

INFORMATICĂ

Culegere de probleme de informatică

Clasele IX-XII

Profil matematică-informatică

Profil informatică intensiv

...e față, care respectă structura programei școlare, se adresează
...or din clasele IX-XII de la liceele cu profil matematică-informatică
...sau informatică intensiv.

...ropuse au grade diferite de dificultate, de la simple probleme de
...prin intermediul cărora elevii își pot consolida partea teoretică,
...eme dificile, care să-i ajute să facă față cu succes examenelor sau
...concursurilor de informatică.

...a, lucrarea este un bun suport pentru profesorii de informatică,
...n material didactic foarte bogat, util pentru buna desfășurare a
...orelor de informatică.

...ect aici!

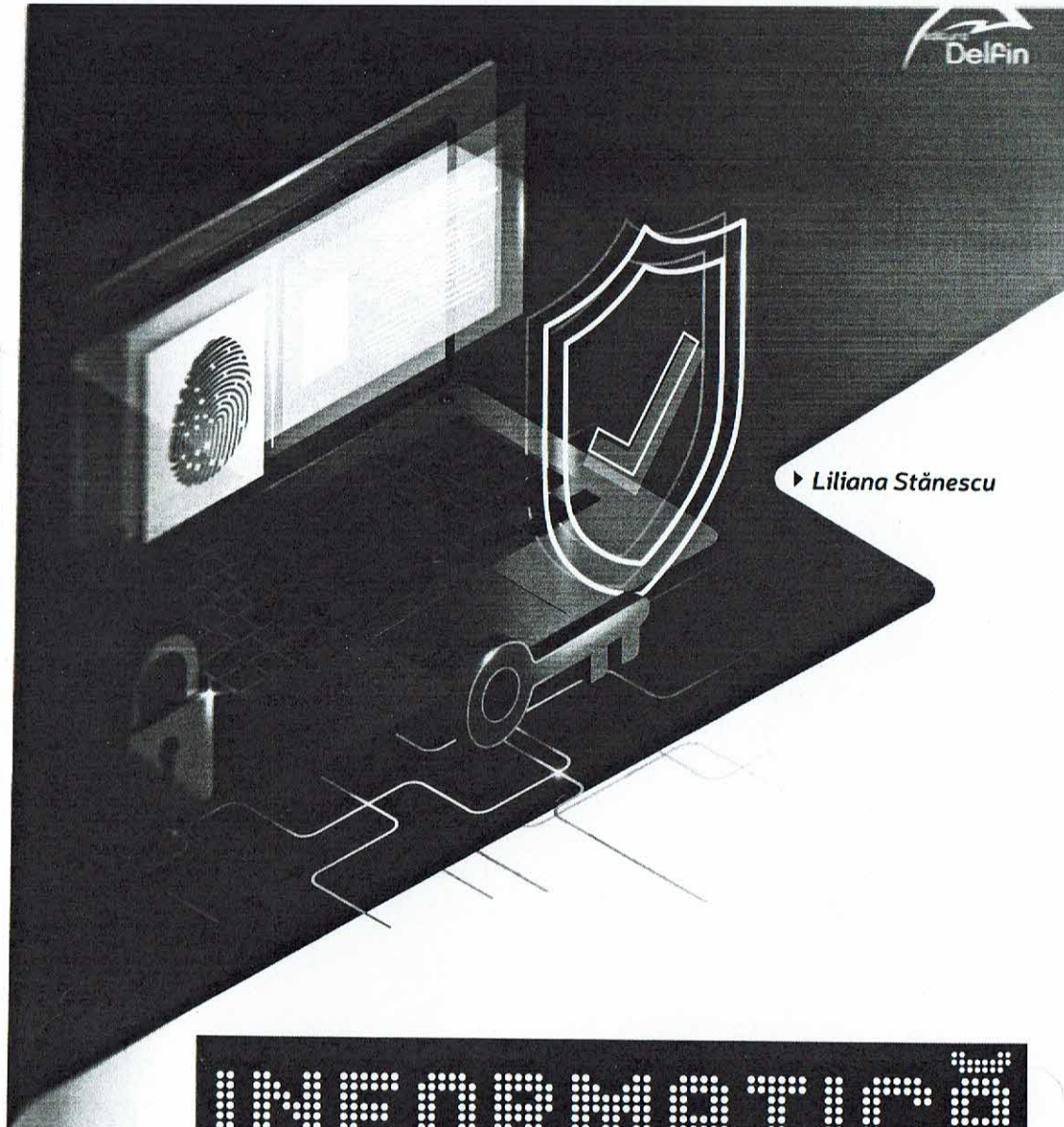


...elfin.ro



9 786069 931196

INFORMATICĂ • Culegere de probleme de informatică • Clasele IX-XII



► **Liliana Stănescu**

INFORMATICĂ

Culegere de probleme de informatică

Clasele IX-XII

Profil matematică-informatică

Lucrarea este realizată în conformitate cu programa școlară în vigoare.

Consilier editorial: Dr. Clementina Ivan-Ungureanu

Coperta colecției: Adina Crăciun

Credite ilustrații: shutterstock.com. / freepik.com

Descrierea CIP a Bibliotecii Naționale a României

STĂNESCU, LILIANA INFORMATICĂ

INFORMATICĂ : Culegere de probleme de informatică / Liliana

Stănescu. - București : Delfin, 2019

ISBN 978-606-993-119-6

51

Editura Delfin, București

Comenzi și distribuție: 021 627 37 20;

0766 22 20 21; 0727 44 44 55

E-mail: contact@edituradelfin.ro

www.edituradelfin.ro

Copyright © Editura Delfin, 2019

Toate drepturile asupra acestei cărți aparțin Editurii Delfin. Reproducerea integrală sau parțială a textului sau a ilustrațiilor din această carte este interzisă fără acordul prealabil scris al Editurii Delfin.

CUPRINS

1. Elemente de bază ale limbajului	7
2. Structuri repetitive	11
2.1. Calcul sume și produse	11
2.2. Divizori. Numere prime	13
2.3. Descompunere în factori primi	16
2.4. Cifre	17
2.5. Operații cu cifre	22
2.6. Baze de numerație	25
2.7. Cel mai mare divizor comun. Cel mai mic multiplu comun	26
2.8. Puterile lui n	27
2.9. Generări de șiruri	28
2.10. Șirul lui Fibonacci	31
2.11. Maxime și minime	31
2.12. Probleme olimpiadă	33
3. Tipuri structurate de date	34
3.1. Tablouri unidimensionale (Vectori)	34
3.1.1. Parcurgeri	34
3.1.2. Construcții de vectori	37
3.1.3. Operații cu vectori	38
3.1.4. Perechi de numere. Triplete de numere.	42
3.1.5. Verificarea unei proprietăți	43
3.1.6. Maxime și minime	44
3.1.7. Sortări	46
3.1.8. Subșiruri	48
3.1.9. Vectori caracteristici	50
3.2. Tablouri bidimensionale (Matrice)	51
3.2.1. Parcurgeri	51
3.2.2. Construcții matrice și vectori	54
3.2.3. Operații cu matrice	59
3.2.4. Diagonale	60
3.2.5. Căutări de elemente	64
3.2.6. Verificare proprietate	65
3.2.7. Maxime și minime	66
3.2.8. Sortări și interclasări	67
3.3. Șiruri de caractere	69
3.3.1. Parcurgeri	69
3.3.2. Operații cu șiruri de caractere	71
3.3.3. Verificare proprietate	74

3.3.4. Construcții cuvinte, șiruri de caractere	74
3.3.5. Cuvinte și fraze	75
3.4. Tipul mulțime	79
3.5. Tipul înregistrare	79
3.6. Fișiere	81
3.7. Probleme olimpiadă	92
4. Subprograme	119
5. Recursivitate	127
5.1. Subprograme recursive	127
5.2. Recursivitate cu vectori	130
5.3. Recursivitate cu matrice	132
5.4. Recursivitate cu șiruri de caractere	133
5.5. Recursivitate indirectă	133
6. Metode de programare	134
6.1. Divide et Impera	134
6.2. Backtracking	137
6.3. Backtracking în plan	142
6.4. Metoda Greedy	143
6.5. Programarea dinamică	147
6.6. Probleme olimpiadă	150
7. Grafuri	161
7.1. Grafuri neorientate	161
7.1.1 Noțiuni generale	161
7.1.2. Grafuri neorientate asociate unui graf dat	165
7.1.3. Lanțuri și cicluri	166
7.1.4. Parcurgerea grafurilor neorientate	168
7.1.5. Conexitate. Componente conexe	171
7.1.6. Tipuri de grafuri speciale	172
7.2. Grafuri orientate (Digrafuri)	175
7.2.1. Noțiuni generale	175
7.2.2. Grafuri orientate asociate unui graf dat	179
7.2.3. Drumuri și circuite	180
7.2.4. Tare conexitate. Componente tare conexe	182
7.2.5. Tipuri speciale de grafuri	183
7.3. Grafuri cu costuri	184
7.4. Probleme olimpiadă	186
8. Arbori	196
8.1. Arbori cu rădăcină	196
8.2. Arbori binari alocați dinamic. Arbore parțial de cost minim	199
8.3. Probleme olimpiadă	202

9. Structuri de date	205
9.1. Stiva	205
9.2. Coada	206
10. Liste alocate dinamic	207
10.1. Liste simplu înlanțuite	207
10.1.1. Parcurgeri	207
10.1.2. Construcție liste	210
10.1.3. Operații cu liste	211
10.2. Liste circulare	213
10.3. Liste dublu înlanțuite	214
Bibliografie	216

1. ELEMENTE DE BAZĂ ALE LIMBAJULUI

1. Să se calculeze suma a două numere a și b date.
2. Se dau 2 numere naturale. Calculați diferența lor.
3. Să se scrie un program care citește de la tastatură 4 note ale unui elev și determină media lor aritmetică.
4. Să se scrie un program care citește de la tastatură un număr natural n și verifică dacă numărul este par sau impar.
5. Fiind dată nota unui elev să se afișeze dacă acesta este *corigent* sau *promovat*.
6. Să se verifice dacă un număr c aparține intervalului $[a, b]$, unde a, b, c sunt date.
7. Afișați ultima cifră pentru suma a două numere naturale.
8. Afișați numărul format din ultimele două cifre ale unui număr n dat de la tastatură.
9. Să se scrie un program care citește de la tastatură un număr întreg și afișează dacă este pozitiv sau negativ.
10. Să se scrie un program care citește de la tastatură două numere întregi și verifică dacă cele două numere au aceeași paritate.
11. Să se scrie un program care determină minimul a trei numere întregi date.
12. Să se scrie un program care citește de la tastatură trei numere naturale și determină diferența dintre cel mai mare și cel mai mic.
13. Fiind date vârstele a patru copii, afișați care dintre ei este cel mai mic și ce diferență de vârstă este între ei.
14. Se citește de la tastatură un număr natural de 3 cifre. Să se stabilească dacă are toate cifrele egale.
15. Se citește de la tastatură un număr natural de 4 cifre. Să se determine câte cifre pare conține.
16. Să se scrie un program care citește de la tastatură un număr natural cu 3 cifre și determină suma cifrelor lui.
17. Scrieți un program care calculează suma pătratelor cifrelor unui număr natural de trei cifre citit de la tastatură.
18. Să se scrie un program care citește de la tastatură un număr natural cu exact 5 cifre și determină numărul obținut prin eliminarea cifrelor din mijloc.
19. Se citește de la tastatură un număr natural de 3 cifre. Să se afișeze cifrele sale în ordine crescătoare.

20. Scrieți un program care citește de la tastatură un număr natural de 4 cifre și care afișează numărul obținut prin schimbarea între ele a primei și ultimei cifre.
21. Se citește de la tastatură un număr natural de 3 cifre. Să se afișeze pe ecran cea mai mare cifră a sa.
22. Se dau de la tastatură două numere a și b cu 4 cifre fiecare. Să se afișeze numărul care are suma cifrelor mai mare.
23. Să se scrie un program care verifică dacă un număr natural citit de la tastatură este pătrat perfect.
24. Să se scrie un program care verifică dacă un număr natural citit de la tastatură este cub perfect.
25. Se dă un număr natural n . Să se afișeze rădăcina sa pătrată.
26. Să se scrie un program care determină aria totală și volumul unui cub pentru care se cunoaște lungimea laturii.
27. Să se scrie un program care citește trei numere reale. Să se permute circular spre dreapta cu două poziții cele trei numere.
28. Se dau trei numere naturale a, b, c . Să se determine cea mai mică valoare care se poate obține prin înmulțirea a două dintre numere și adunarea rezultatului cu al treilea.
29. Se citesc de la tastatură trei numere naturale. Să se determine câte dintre ele sunt distincte.
30. Se citesc de la tastatură trei numere a, b, c . Să se verifice dacă cele trei numere sunt în ordine crescătoare. În caz contrar, să se afișeze în ordine crescătoare.
31. Se citește un număr natural n . Să se stabilească dacă n poate fi scris ca sumă de trei numere naturale consecutive.
32. Se citește un număr natural n . Să se stabilească dacă n poate fi scris ca produs de două numere naturale consecutive.
33. Să se calculeze aria unui triunghi, cunoscând laturile a, b, c ale triunghiului.
34. Fie a și b două variabile întregi. Să se interschimbe valorile celor două variabile folosind o variabilă auxiliară.
35. Fie a și b două variabile întregi. Să se interschimbe valorile celor două variabile, fără a folosi o variabilă auxiliară.
36. Să se verifice dacă o fracție a/b se simplifică printr-un număr d dat de la tastatură. În caz afirmativ se va afișa fracția simplificată.
37. Se citește de la tastatură o literă din alfabetul englez. Să se verifice dacă este vocală sau consoană.

38. Se dau de la tastatură patru valori a, b, c, d . Să se determine și să se afișeze maximul valorilor pare și minimul valorilor impare dintre cele 4 numere.
39. Se citesc de la tastatură două numere întregi a și b și un operator matematic x care poate fi $+$, $-$, $*$, $/$. Să se afișeze valoarea expresiei axb .
40. Să se verifice dacă trei numere date a, b, c sunt pitagoreice (verifică una din condițiile $a^2 = b^2 + c^2$, $b^2 = a^2 + c^2$, $c^2 = a^2 + b^2$).
41. Se citesc de la tastatură trei numere naturale a, b, c . Să se verifice dacă acestea pot fi unghiuri ale unui triunghi exprimate în grade. În caz afirmativ, să se determine natura triunghiului (ascuțitunghic, dreptunghic, obtuzunghic).
42. Se dau de la tastatură trei numere naturale a, b, c . Să se determine dacă pot reprezenta lungimile laturilor unui triunghi (suma oricăror două este mai mare decât a treia). În caz afirmativ să se precizeze de ce tip este triunghiul: echilateral, isoscel, dreptunghic sau oarecare.
43. Se dau de la tastatură trei numere naturale a, b, c . Să se permute circular spre stânga valorile acestor variabile și să se afișeze.
44. Se dau de la tastatură trei numere naturale a, b, c . Să se permute circular spre dreapta valorile acestor variabile și să se afișeze.
45. Să se rezolve ecuația de gradul al II-lea $ax^2 + bx + c = 0$, unde a, b, c sunt citite de la tastatură. Se cere să se trateze toate cazurile posibile.
46. Să se rezolve sistemul de ecuații:
$$\begin{cases} ax + by = c \\ dx + ey = f \end{cases}$$
, unde a, b, c, d, e, f sunt numere reale nenule. Să se analizeze toate cazurile posibile.
47. Să se scrie un program care convertește următoarele unități de măsură

1. Inch	2. Pound	3. Foot	4. Gallon
---------	----------	---------	-----------

 Astfel 1.Inch=2.54 cm 2.Pound=0.45kg 3.Foot=0.30 m 4.Gallon=3.78l
48. Să se verifice dacă patru numere date sunt în progresie aritmetică. În caz afirmativ să se precizeze rația progresiei.
49. Să se verifice dacă patru numere date sunt în progresie geometrică. În caz afirmativ să se precizeze rația progresiei.
50. Să se verifice dacă patru numere întregi date, pot forma o mulțime (sunt distincte două câte două).
51. Să se calculeze suma și diferența a două unghiuri date u și v în grade minute și secunde.
52. De la tastatură se citește un număr întreg n ($1 \leq n \leq 10$). Elaborați un program prin intermediul căruia se va afișa pe ecran tabla înmulțirii cu numărul n .
53. De la tastatură se citește o cifră c . Să se afișeze cifra scrisă cu caractere.

54. Se consideră patru numere a, b, c, d . Să se verifice dacă pot fi termenii unei progresii aritmetice sau geometrice. În caz afirmativ să se precizeze timpul de progresie și rația.
55. Să se determine ultima cifră a numărului 3^x , unde x este un număr natural dat. Nu se va calcula valoarea lui 3^x .
56. Să se determine ultima cifră a numărului $2^x + 7^x$, unde x este un număr natural dat. Nu se va calcula valoarea sumei.
57. Să se determine ultima cifră a numărului x^y , unde x și y sunt două valori naturale date ($x < 10$). Nu se va calcula valoarea lui x^y .
58. Să se calculeze valoarea funcției $f: R \rightarrow R$, pentru o valoare x dată de la tastatură, unde $f(x) = \begin{cases} x^2 + 3x + 4, & x < 2 \\ x - 7, & x \geq 2 \end{cases}$.
59. Să se calculeze valoarea funcției $f: R \rightarrow R$, pentru o valoare x dată de la tastatură, unde $f(x) = \begin{cases} x^2 - 2x + 3, & x < 1 \\ 2x + 4, & x \geq 1 \end{cases}$.
60. Să se calculeze valoarea funcției $f: R \rightarrow R$, pentru o valoare x dată de la tastatură, unde $f(x) = \begin{cases} 3x + 2, & x < 3 \\ \frac{2x+3}{5x-1}, & 3 \leq x \leq 5 \\ \sqrt{x^2 + x + 1}, & x > 5 \end{cases}$.
61. Să se scrie un program care calculează suma a două intervale de timp, exprimate în ore, minute și secunde. Cele două intervale de timp sunt date de la tastatură în forma precizată mai sus.
62. De la tastatură se citește un număr ce reprezintă o zi calendaristică. Să se afișeze decada căreia îi aparține ziua.
63. Se citește de la tastatură o cifră între 1 și 7, reprezentând o zi din săptămână. Să se afișeze denumirea zilei.
64. Să se verifice dacă un an calendaristic dat este sau nu bisect.
65. Se introduc de la tastatură numele și media generală a trei elevi. Să se afișeze numele acestora, ordonați descrescător după medii.

2. STRUCTURI REPETITIVE

2.1. Calcul sume și produse

1. Să se calculeze suma $S = 1 + 2 + 3 + \dots + n$, unde n este un număr natural dat.
2. Să se calculeze suma $S = 2 + 4 + 6 + \dots + n$, unde n este un număr natural par dat.
3. Să se calculeze suma $S = 1 + 3 + 5 + \dots + n$, unde n este un număr natural impar dat.
4. Să se calculeze suma $S = 1 \cdot 2 + 2 \cdot 3 + \dots + n \cdot (n + 1)$, n este un număr natural dat de la tastatură.
5. Să se calculeze suma $S = 1 \cdot 2 - 2 \cdot 3 + \dots \pm n \cdot (n + 1)$, n este un număr natural dat.
6. Să se calculeze suma $S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n}$, unde n este un număr natural dat de la tastatură.
7. Să se calculeze suma $S = \frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} + \dots + \frac{1}{n \cdot (n+1)}$, unde n este un număr natural dat.
8. Să se calculeze suma $S = 1 \cdot n + 2 \cdot (n - 1) + \dots + n \cdot 1$, unde n este un număr natural dat.
9. Să se calculeze suma $S = 1 - 2 + 3 - 4 + \dots \pm n$, pentru n dat de la tastatură.
10. Calculați produsul $P = \left(1 - \frac{1}{2^2}\right) \left(1 - \frac{1}{3^2}\right) \dots \left(1 - \frac{1}{n^2}\right)$, pentru n dat de la tastatură.
11. Să se calculeze suma $S = \frac{1}{2} + \frac{2}{3} + \dots + \frac{n-1}{n}$, unde n este un număr natural dat.
12. Să se calculeze suma $S = 1^2 + 2^2 + 3^2 + \dots + n^2$, n este un număr natural dat de la tastatură.
13. Să se calculeze suma $S = 1^3 + 2^3 + 3^3 + \dots + n^3$, unde n este un număr natural dat.
14. Să se calculeze suma $S = 1^4 + 2^4 + 3^4 + \dots + n^4$, n este un număr natural dat.
15. Se citește un număr natural n . Să se calculeze suma $S = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{n^2}$.
16. Să se calculeze suma $S = 1 + 1 \cdot 2 + 1 \cdot 2 \cdot 3 + \dots + 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$, unde n este un număr natural dat de la tastatură.

17. Să se calculeze suma $S = 1 - 1 \cdot 2 + 1 \cdot 2 \cdot 3 \dots \dots \dots \pm 1 \cdot 2 \cdot 3 \dots n$, unde n este un număr natural dat.
18. Să se calculeze suma $S = \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \dots + \frac{1}{1 \cdot 2 \cdot 3 \dots n}$, n este un număr natural dat.
19. Să se calculeze suma $S = 1^2 + 4^2 + 7^2 + \dots \dots \dots (3n - 2)^2$, pentru n dat de la tastatură.
20. Să se calculeze suma $S = 1 + 2^2 + 3^3 + \dots + n^n$, pentru n dat de la tastatură.
21. Să se calculeze produsul $P = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$, unde n este un număr natural dat.
22. Să se calculeze produsul $P = 2 \cdot 4 \cdot 6 \cdot \dots \cdot n$, n este un număr natural par dat.
23. Să se calculeze suma $S = 1! + 2! + 3! + \dots + n!$, unde n este un număr natural dat.
24. Să se calculeze produsul $P = \frac{3}{4} \cdot \frac{8}{9} \cdot \frac{15}{16} \cdot \dots \cdot \frac{n(n+2)}{(n+1)^2}$, pentru un număr n dat de la tastatură.
25. Să se calculeze suma $S = \frac{1}{2} + \frac{2}{2^3} + \frac{3}{2^5} + \dots + \frac{n}{2^{2n-1}}$, pentru n dat de la tastatură.
26. Să se scrie un program care citește de la tastatură o valoare x și calculează valoarea expresiei:
- $$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots + (-1)^n \frac{x^{2n}}{(2n)!}$$
27. Un număr natural nenul n se numește *norocos* dacă pătratul lui se poate scrie ca sumă de n numere naturale consecutive. Se da un număr natural n . Dacă numărul este norocos afișați cele n numere consecutive care adunate dau pătratul acestuia (Exemplu 3 și 7 sunt numere norocoase; $3^2 = 2 + 3 + 4$; $7^2 = 4 + 5 + \dots + 10$).
28. Se dau $2n$ numere naturale, unde n este număr natural. Să se calculeze suma produselor dintre fiecare număr din prima jumătate și fiecare număr din a doua jumătate a șirului de numere (Suma este de forma $S = a_1 \cdot a_{n+1} + a_2 \cdot a_{n+2} + \dots + a_n \cdot a_{2n}$).
29. Să se afle suma resturilor împărțirii tuturor numerelor naturale de la 1 la n printr-un număr k .
30. Într-un șir de numere naturale se numește *vârf* un element care are doi vecini și este strict mai mare decât aceștia. Se da un șir cu n elemente, numere naturale. Calculați suma elementelor din șir care sunt vârfuri.
31. Fiind dat un număr natural n , să se determine pentru fiecare număr x de la 1 la n cel mai mare rest care se poate obține împărțind numărul x la toate numerele de la 1 la x . Să se calculeze suma acestor resturi și să se afișeze.

32. Se citește de la tastatură un număr natural n . Să se scrie n în toate modurile posibile, ca sumă de numere întregi consecutive.
33. *Înmulțirea a la russe*. Fiind date două numere naturale, să se determine produsul lor după următorul exemplu:

- Considerăm numerele $36 \cdot 23 = 828$. Deînmulțitul (36) și înmulțitorul (23) își modifică valorile astfel: deînmulțitul se împarte succesiv la 2, iar înmulțitorul se înmulțește succesiv cu 2, până când deînmulțitul devine 1
- Produsul celor două numere se află adunând înmulțitorul de pe liniile pentru care deînmulțitul este impar

Deînmulțitul	Înmulțitorul	Se adună
36	23	-
18	46	-
9	92	92
4	184	-
2	368	-
1	736	736

$$P = 92 + 736 = 828$$

2.2. Divizori. Numere prime.

- Să se afișeze toate numerele mai mici decât un n dat, care sunt divizibile cu 5.
- Se citește un număr natural n . Să se afișeze divizorii numărului citit.
- Se citește un număr natural n . Să se determine suma divizorilor săi.
- Se citește un număr natural n . Să se determine suma divizorilor săi impari.
- Se citește un număr natural n . Să se determine câți divizori are.
- Se citește un număr natural n . Să se determine câți divizori pari are acest număr.
- Se citește de la tastatură un număr natural n . Să se calculeze și să se afișeze media aritmetică a tuturor divizorilor săi.
- Se citește un număr natural n . Să se determine numărul de divizori ai oglinditului lui n .
- Se citesc două numere a și b ($a < b$) având cel mult 9 cifre fiecare. Afișați câte numere din intervalul $[a, b]$ au exact 3 divizori.
- Să se afișeze numerele mai mici sau egale decât un număr n care au k divizori (n, k sunt date de la tastatură).
- Se citesc două numere naturale n și k . Afișați numerele naturale mai mici sau egale decât n care au cel puțin k divizori impari.

12. Să se scrie un program care determină cel mai mic număr natural care are k divizori (k este citit de la tastatură).
13. Să se scrie un program care afișează divizorii comuni a două numere naturale citite de la tastatură.
14. Se citesc două numere a și b . Afișați numerele din intervalul $[a, b]$ cu proprietatea că au număr maxim de divizori.
15. Un număr natural nenul se numește *sPar* dacă atât el, cât și suma divizorilor săi proprii (divizori diferiți de 1 și de el însuși), sunt numere pare. Se citește un număr natural n ($n \in [1, 10^9]$) și se cere să se scrie cel mai mic număr *sPar*, strict mai mare decât n . (Examen de Bacalaureat - Simulare 2016)
16. Se citesc două numere a și b . Afișați numerele din intervalul $[a, b]$ cu proprietatea că au număr maxim de divizori primi.
17. Se citește un număr natural n ($n \geq 2$). Să se scrie numerele x, y și z , divizori ai lui n , cu proprietatea că $0 < x < y < z$ și $x + y + z = n$. Numerele se scriu în ordine strict crescătoare, separate prin câte un spațiu, iar dacă n nu are astfel de divizori, se scrie mesajul *Nu există*.
18. Un număr se numește *perfect* dacă este egal cu suma divizorilor săi mai mici decât el. Să se verifice dacă un număr n dat este perfect.
19. Să se determine numerele perfecte mai mici sau egale decât un număr n dat de la tastatură.
20. Un număr natural nenul se numește *subperfect* dacă este strict mai mic decât suma divizorilor săi proprii (divizori naturali diferiți de 1 și de el însuși). *Exemplu:* 12 este număr subperfect pentru că $12 < 2+3+4+6$. Se citesc două numere naturale a și b ($2 \leq a < b \leq 10^9$) și se cere să se afișeze, separate prin câte un spațiu, în ordine descrescătoare, toate numerele subperfecte din intervalul $[a, b]$. Dacă în interval nu există astfel de numere, se afișează mesajul *Nu există*. (Examen de Bacalaureat - Model 2015)
21. Să se verifice dacă un număr natural n dat este *pătrat perfect*.
22. Să se afișeze toate numerele din intervalul $[1, n]$ care sunt pătrate perfecte.
23. Să se afișeze toate pătratele perfecte mai mici decât o valoare n dată de la tastatură.
24. Se citește de la tastatură un număr natural n . Să se verifice dacă numărul dat este *prim* (un număr este prim dacă are decât 2 divizori, 1 și el însuși).
25. Să se afișeze toate numerele prime mai mici sau egale decât un n dat de la tastatură.
26. Să se scrie un program care afișează primele n numere prime.
27. Să se afișeze numerele prime din intervalul $[a, b]$ dat.
28. Să se afișeze toate numerele prime de 3 cifre.

29. Să se afișeze numerele prime de 3 cifre care citite invers sunt tot numere prime.
30. Să se afișeze toate numerele prime de p cifre, p este un număr natural dat.
31. Se citesc două numere naturale a și b . Calculați și afișați câte numere din intervalul $[a, b]$ au proprietatea că sunt încadrate de două numere prime.
32. Se citește un număr natural n și apoi n numere naturale. Să se determine câte dintre cele n numere citite au proprietatea că au numărul de divizori egal cu poziția pe care au fost citite.
33. Se citește un număr natural k de o cifră. Afișați toate numerele n formate din k cifre care au proprietatea că $n - 1$ și $n + 1$ sunt numere prime.
34. Se citește un număr natural n . Afișați toate perechile de numere prime consecutive mai mici sau egale decât n .
35. Se citește un număr natural n . Afișați primele n perechi de numere prime consecutive.
36. Să se afișeze toate numerele naturale din intervalul $[a, b]$ care au un număr prim de divizori.
37. Se citesc două numere naturale a și b . Afișați toate perechile de numere x și y din intervalul $[a, b]$ care au proprietatea că au același număr de divizori.
38. Se citește de la tastatură un număr natural n . Să se gasească cel mai apropiat număr prim de n (diferența în modul între n și numărul prim găsit trebuie să fie minimă).
39. Se citește un număr natural, n ($n \geq 2$) și se cere să se scrie cel mai mic număr natural care are aceiași divizori primi ca n . (Examen de Bacalaureat - Sesiunea Specială 2016)
40. Un număr natural n se numește *superprim* dacă atât el cât și toate prefixele sale sunt numere prime. Să se verifice dacă un număr n dat este superprim.
41. Se dă un șir cu n numere naturale. Să se afișeze numerele prime din șir.
42. Două numere naturale a și b se numesc *prietene* dacă suma divizorilor lui a (fără a) este egală cu b și suma divizorilor lui b (fără b) este egală cu a . Să se verifice dacă 2 numere date a și b sunt prietene.
43. Să se afișeze toate numerele mai mici sau egale cu n care au proprietatea că atât ele cât și răsturnatele lor sunt numere prime.
44. Se consideră un șir cu n numere naturale. Determinați suma celor care au oglinditul număr prim.
45. Se consideră un șir cu n numere naturale. Determinați câte dintre ele sunt prime și au suma cifrelor un număr prim.
46. Să se determine cel mai mic număr prim mai mare decât un număr natural n citit de la tastatură.
47. Se citește un șir de numere întregi până când se introduce de două ori consecutiv aceeași valoare. Să se afișeze cât numere prime sunt în șir.

48. Să se scrie un program care citește de la tastatură un șir de n numere naturale și determină media aritmetică a celor prime.
49. Se citește un număr natural. Să se verifice dacă este *aproape prim*. (Un număr natural se numește *aproape prim* dacă poate fi scris ca produs de două numere naturale prime distincte.)
50. Se citește un număr natural n și apoi n numere naturale. Să se determine câte dintre ele sunt *aproape prime*.
51. Se citește un număr natural n și apoi n numere naturale. Să se determine cel mai mare număr *aproape prim* dintre cele date.
52. Se dau n numere naturale. Să se afișeze cel mai mare, respectiv cel mai mic număr prim dintre cele n numere citite. Se consideră ca șirul are cel puțin două numere prime.
53. Se dau n numere naturale. Calculați suma obținută prin adunarea celui mai mare divizor prim al fiecărui număr dat.
54. Orice număr natural par mai mare decât 4 se poate scrie ca sumă de 2 numere prime impare (*conjectura lui Goldbach*). Să se verifice această proprietate pentru numerele de 3 cifre.
55. Să se descompună un număr natural n în toate modurile ca sumă de două numere prime. Dacă nu exista nici o descompunere, să se afișeze mesajul *Nu există*.
56. Se citesc de la tastatură n numere naturale. Să se determine cu câte zerouri se termină produsul acestora, fără a efectua produsul.
57. Se citesc de la tastatură n numere naturale și un număr prim p . Să se determine un număr k (k maxim) astfel încât p^k să dividă produsul celor n numere naturale. Nu se va efectua produsul celor n numere.

2.3. Descompunere în factori primi

1. Să se descompună în factori primi un număr natural n .
2. Să se descompună în factori primi un număr natural n și să se afișeze factorii și puterile la care apar în descompunere.
3. Se citește un număr natural n . Afișați factorul prim care apare la puterea cea mai mare în descompunerea în factori primi a lui n .
4. Se citește un număr natural n . Să se afișeze factorii primi ai lui n în ordine crescătoare.
5. Se dau n numere naturale nenule. Determinați cel mai mare număr prim care apare în descompunerea în factori a produsului celor n numere, precum și exponentul său în descompunere.
6. Se citesc două numere naturale a și b . Afișați numărul care are suma maximă a exponenților din descompunerea în factori primi. Dacă ambele numere au aceeași sumă se va afișa unul dintre ele.

7. Se citesc două numere naturale a și b . Afișați numărul care are cei mai mulți factori primi. Dacă ambele numere au același număr de factori primi, se va afișa unul dintre ele.
8. Să se scrie un program care citește un șir de n numere naturale și determină numărul din șir care are număr maxim de factori primi.
9. Se dau două numere naturale nenule. Să se verifice dacă cele două numere au exact aceiași factori primi, indiferent de puterea acestora.
10. Se dau două numere prime p, q și n numere naturale nenule. Determinați exponentul maxim e pentru care numărul $p^e \cdot q^e$ divide produsul celor n numere date.
11. Se citește un număr natural n . Să se determine numărul de zerouri de la sfârșitul scrierii zecimale a numărului $n!$.
12. Se dau n numere naturale nenule. Calculați numărul de zerouri de la sfârșitul scrierii zecimale a produsului celor n numere. Nu se va calcula produsul celor n numere.
13. Se dau n numere naturale nenule. Afișați ultima cifră nenulă din scrierea zecimală a produsului celor n numere.
14. Să se determine exponentul la care apare numărul prim p în descompunerea numărului $1 \cdot 2 \cdot 3 \cdot \dots \cdot n$ fără a calcula înmulțirea.
15. Să se scrie un program care, pentru un număr n citit de la tastatură, afișează descompunerea în factori primi ai lui $n!$.

2.4. Cifre

1. Să se scrie un program care să determine suma cifrelor unui număr natural citit de la tastatură.
2. Să se scrie un program care să determine suma cifrelor pare ale unui număr natural citit de la tastatură.
3. Să se scrie un program care să determine produsul cifrelor impare ale unui număr natural citit de la tastatură.
4. Să se scrie un program care să determine numărul de cifre ale unui număr natural citit de la tastatură.
5. Să se scrie un program care să determine numărul de cifre impare ale unui număr natural citit de la tastatură.
6. Se citește de la tastatură un număr natural n . Să se determine de câte ori apare cifra c printre cifrele numărului dat.
7. Se citește de la tastatură un număr natural n . Să se determine de câte ori apare cifra 0 printre cifrele numărului dat.
8. Se consideră un număr natural n . Calculați produsul primelor două cifre ale sale.

9. Să se scrie un program care să determine suma primelor trei cifre ale unui număr natural citit de la tastatură (se consideră că numărul introdus are cel puțin 4 cifre).
10. Se citește un număr natural n . Să se construiască și să se afișeze *oglinzitul* numărului citit (oglinzitul unui număr se obține citind numărul inițial, de la dreapta la stânga).
11. Se citește un număr natural n . Calculați și afișați oglinzitul sumei cifrelor lui n .
12. Să se scrie un program care să determine cea mai mare cifră a unui număr natural citit de la tastatură.
13. Să se scrie un program care să determine cea mai mică cifră a unui număr natural citit de la tastatură.
14. Se citește de la tastatură un număr natural. Să se determine cea mai mare cifră care apare în scrierea acestuia și numărul de apariții ale ei.
15. Să se scrie un program care să determine cea mai mare cifră impară a unui număr natural citit de la tastatură și de câte ori apare în scrierea numărului.
16. Să se scrie un program care să determine suma dintre cea mai mică și cea mai mare cifră ale unui număr natural citit de la tastatură.
17. Să se scrie un program care să determine ultima cifră pară a unui număr natural citit de la tastatură (numărarea pozițiilor se face începând cu cifra unităților – poziția 1).
18. Să se scrie un program care să determine prima cifră, număr prim, a unui număr natural citit de la tastatură (numărarea pozițiilor se face începând cu cifra unităților – poziția 1).
19. Se dă un număr natural format din cifrele 1 sau 2. Aflați cifra care apare de cele mai multe ori în scrierea numărului.
20. Să se afișeze toate numere de 4 cifre care conțin numărul 22 la început.
21. Să se afișeze toate numere de 4 cifre care conțin numărul 22 la mijloc.
22. Se citește un număr natural n . Să se determine suma cifrelor aflate pe poziții impare (numărarea pozițiilor în numărul dat se face începând de la cifra unităților - poziția 1).
23. Să se scrie un program care să determine numărul de apariții a ultimei cifre în scrierea unui număr natural citit de la tastatură (numărarea pozițiilor se face începând de la cifra unităților - poziția 1).
24. Să se scrie un program care să determine numărul de apariții a primei cifre în scrierea unui număr natural citit de la tastatură (numărarea pozițiilor se face începând de la cifra unităților - poziția 1).
25. Se citește un număr natural n . Să se determine suma cifrelor aflate pe poziții pare. Numărarea pozițiilor în numărul dat se face începând de la cifra unităților (poziția 1).
26. Se citește un număr natural, n și se cere să se scrie numărul de cifre prime

27. Se citesc două numere naturale n și m . Să se decidă dacă cele două numere au cel puțin o cifră comună.
28. Se citesc două numere a și b , fiecare având cifrele distincte. Determinați câte cifre comune au cele 2 numere.
29. Se dau două numere naturale. Dacă cele două numere au același număr de cifre, să se afle pe câte poziții aceste cifre coincid.
30. Se citește un număr natural n și se cere să se scrie numărul cifrelor care apar o singură dată în scrierea lui n . (Examen de Bacalaureat - Model 2016)
31. Să se afișeze cifra care apare de cele mai multe ori într-un număr natural n și de câte ori apare ea în n .
32. Se citește un număr natural n , se cere să se afișeze toate cifrele distincte care apar în scrierea lui n , separate prin câte un spațiu. (Examen de Bacalaureat – Simulare 2015)
33. Se citește de la tastatură un număr natural n . Să se verifice dacă numărul citit are toate cifrele egale.
34. Numim *urma unui număr natural* x , numărul obținut prin înlocuirea fiecărei cifre cu complementul său în baza 10 (complementul cifrei c este $9 - c$). Să se scrie un program care returnează urma unui număr natural x citit de la tastatură.
35. Să se verifice dacă pentru un număr natural n dat, urma lui n coincide cu oglinzitul numărului inițial.
36. Se citește un număr natural nenul n . Să se construiască două numere n_1 și n_2 , astfel încât numărul n_1 să fie format doar din cifrele pare ale lui n , iar numărul n_2 doar din cifrele impare ale lui n . Numerele se vor forma cu cifrele în ordinea inversă a apariției lor în numărul n . Calculați suma celor 2 numere.
37. Se citește un număr natural nenul n . Să se construiască două numere n_1 și n_2 , astfel încât numărul n_1 să fie format doar din cifrele pare ale lui n , iar numărul n_2 doar din cifrele impare ale lui n . Numerele se vor forma cu cifrele în ordinea apariției lor în numărul n . Afișați cele 2 numere.
38. Fie n un număr natural. Aflați cel mai mare număr care are același număr de cifre cu n și suma cifrelor egale cu suma cifrelor lui n .
39. Un număr natural se numește *autopomorfic*, dacă el este sufixul pătratului său. Să se verifice dacă un număr x dat este autopomorfic (25 este autopomorfic, deoarece $25^2 = 625$, deci 25 se găsește ca sufix în numărul 625).
40. Să se determine toate numere autopomorfe dintr-un interval $[a, b]$ dat.
41. Determinați numărul de apariții a unei cifre c în reprezentarea tuturor numerelor mai mici sau egale cu un n dat.
42. Se consideră un interval $[a, b]$. Afișați câte dintre numerele din intervalul $[a, b]$ au proprietatea că suma cifrelor lor este divizibilă cu un număr k dat.

43. Se dă un interval $[a, b]$. Afișați câte dintre numerele din intervalul $[a, b]$ au proprietatea că atât ele cât și oglinditele lor sunt pătrate perfecte (144 și 441 îndeplinesc condiția).
44. Se dau două numere naturale a și b . Determinați câte numere naturale din intervalul $[a, b]$ se divid cu toate cifrele lor nenule.
45. Se citesc două numere naturale, a și b ($10 \leq a \leq b \leq 10^6$) și se cere să se scrie toate numerele naturale din intervalul $[a, b]$ care au toate cifrele identice. Numerele scrise sunt separate prin câte un spațiu, iar dacă nu există astfel de numere, se scrie mesajul *Nu există*. (Examen de Bacalaureat – Sesiunea Specială 2017)
46. Să se determine toate numerele naturale de trei cifre cu proprietatea că suma inverselor cifrelor lor este un număr subunitar, iar cifrele sunt în ordine strict crescătoare.
47. Se introduc numere naturale până când se introduce numărul 0. Afișați suma obținută prin adunarea primei și a ultimei cifre din fiecare număr citit. Numerele cu mai puțin de două cifre nu se iau în considerare.
48. Se citește de la tastatură un număr natural n , apoi n numere naturale. Să se calculeze suma obținută prin adunarea cifrei din mijloc a fiecărui număr care are număr impar de cifre.
49. Se citește de la tastatură un număr natural n , apoi n numere naturale. Să se afișeze numărul format din prima cifră a fiecărui număr (prima cifră se consideră cifra unităților).
50. Se citește de la tastatură un număr natural n , apoi n numere naturale. Să se calculeze suma obținută prin adunarea ultimei cifre a fiecărui număr (prima cifră se consideră cifra unităților).
51. Se citește de la tastatură un număr natural n , apoi n numere naturale. Să se afișeze numărul format din ultima cifră a fiecărui număr (prima cifră se consideră cifra unităților).
52. Se citesc n numere naturale. Determinați câte perechi de numere citite consecutiv au aceeași sumă a cifrelor.
53. Să se afișeze toate perechile de numere mai mici sau egale cu n care au aceeași sumă a cifrelor.
54. Se citește o cifră k și apoi se citesc numere până la apariția lui 0. Să se determine de câte ori apare cifra k în numerele citite.
55. Se citesc numere naturale până când se introduce valoarea 0. Afișați suma obținută prin adunarea numerelor formate din primele două cifre ale numerelor citite. Nu se iau în considerare numerele cu mai puțin de 2 cifre.
56. Se citesc n numere naturale. Determinați câte perechi de numere, citite consecutiv au sumele cifrelor de aceeași paritate.
57. Se citesc perechi de numere naturale până la citirea a două valori nule. Să se determine câte dintre perechi încep cu aceeași cifră.

58. Să se determine *cifra de control* a unui număr n dat. (*Cifra de control* a unui număr n se obține calculând repetat suma cifrelor lui n și înlocuindu-l pe n cu suma calculată, până se obține o singură cifră).
59. Se dau n numere naturale. Afișați numărul pentru care diferența în valoare absolută dintre el și oglinditul său este minimă.
60. Se citesc numere naturale până când se introduce valoarea 0. Pentru fiecare număr citit, calculați suma cifrelor pare și suma cifrelor impare.
61. Se citește un număr natural n . Să se determine dacă are cifrele ordonate crescător.
62. Se citește un număr natural n . Să se determine dacă are cifrele ordonate descrescător.
63. Se citește de la tastatură un număr natural n . Să se verifice dacă numărul citit are *aspect de munte* (cifrele sunt în ordine crescătoare până la o poziție și în ordine descrescătoare de la acea poziție până la sfârșit).
64. Se citește de la tastatură un număr natural n . Să se verifice dacă numărul citit are *aspect de vale* (cifrele sunt în ordine descrescătoare până la o poziție și în ordine crescătoare de la acea poziție până la sfârșit).
65. Se citesc numere naturale până când se introduce valoarea 0. Pentru fiecare număr citit, calculați numărul format din cifrele lui pare, numărul format din cifrele lui impare și afișați-l pe cel mai mare din fiecare pereche.
66. Să se verifice dacă un număr n dat este *palindrom* (citit de la stânga la dreapta sau de la dreapta la stânga reprezintă același număr).
67. Se citesc două numere naturale a și b . Afișați numerele *palindrom* din intervalul $[a, b]$.
68. Se citesc două numere naturale a și b . Afișați cel mai mic număr *palindrom* din intervalul $[a, b]$.
69. Calculați și afișați media aritmetică a palindroamelor din intervalul $[a, b]$.
70. Se citește de la tastatură un număr natural n . Să se determine cel mai apropiat număr palindrom față de n (diferența absolută între numărul n și palindromul găsit este minimă).
71. Numim număr de tip *dublu palindrom*, o valoare de tip palindrom cu număr par de cifre în care fiecare cifră de rang impar este egală cu cifra alăturată din dreapta ei. Să se determine numerele de 4 cifre de tip dublu palindrom.
72. Se citește un număr natural n și apoi n numere naturale. Afișați numerele care au suma cifrelor egală cu suma cifrelor primului număr citit.
73. Se citește un număr natural n și apoi n numere naturale. Afișați câte dintre numerele citite au răsturnatul egal cu primul număr citit.
74. Se citește un număr natural n și apoi n numere naturale. Afișați câte dintre ele au numărul cifrelor egal cu numărul lor de ordine de la citire.

75. Se citește un număr natural n cu număr par de cifre. Calculați și afișați numărul obținut din n inversând cifra unităților cu a zecilor, cea a sutelor cu cea a miilor etc.
76. Se citește un număr natural n . Afișați cele două numere obținute prin împărțirea la mijloc a numărului n . Se consideră că numărul n este format dintr-un număr par de cifre.
77. Se citesc numere naturale până când se introduce valoarea 0. Afișați suma obținută prin adunarea numerelor formate din cifra (cifrele) din mijloc ale numerelor citite (dacă numărul citit are un număr impar de cifre se va lua în considerare cifra din mijloc, iar dacă numărul are un număr par de cifre se va lua numărul de două cifre din mijloc).
78. Se citesc două numere a și b și o cifră c . Calculați de câte ori apare cifra c în scrierea numerelor naturale din intervalul $[a, b]$.
79. Se citește de la tastatură un număr natural n . Să se afișeze inversul numărului n cu două zecimale exacte.
80. Se citesc de la tastatură două numere naturale nenule m și n . Să se scrie fracția $1/m$ ca fracție zecimală, cu n zecimale.
81. *International Standard Book Number (ISBN)* este un cod unic utilizat pentru identificarea fiecărei cărți, format din 9 cifre urmate de o cifră de control care poate fi o cifră sau caracterul x ce reprezintă numărul 10. Semnificația cifrelor din cadrul unui cod ISBN este de forma $\overline{a_1 a_2 \dots a_{10}}$ este următoarea: $\overline{a_1 a_2 a_3}$ identifică țara, $\overline{a_4 a_5 a_6 a_7}$ identifică editura, $\overline{a_8 a_9}$ identifică numărul asociat cărții, iar a_{10} reprezintă cifra de control calculată astfel încât $\sum_{i=1}^{10} a_i \cdot i$ să fie divizibilă cu 11. De exemplu 973 8934 05 2 este un cod ISBN cu cifra de control 2. Scrieți un program care citește de la tastatură un număr natural cu 9 cifre care reprezintă primele 9 poziții ale unui cod ISBN și afișează pe ecran cifra de control corespunzătoare. (Variante Bacalaureat 2009)

2.5. Operații cu cifre

1. Se citește de la tastatură un număr natural n . Să se afișeze numărul obținut prin eliminarea primei cifre a numărului citit.
2. Se citește de la tastatură un număr natural n . Să se afișeze numărul obținut prin eliminarea ultimei cifre a numărului citit.
3. Se citește de la tastatură un număr natural n . Să se afișeze numărul obținut prin eliminarea cifrei din mijloc a numărului citit, dacă acesta are un număr impar de cifre și prin eliminarea celor 2 cifre din mijloc, dacă acesta are un număr par de cifre.
4. Se citește de la tastatură un număr natural n . Să se afișeze numărul obținut prin eliminarea cifrei de pe poziția k a numărului citit (cifra unităților se află pe prima poziție în număr).

5. Se citesc un număr natural n și o cifră c . Eliminați toate aparițiile cifrei c din numărul n și afișați numărul astfel obținut.
6. Se citește un număr natural cu toate cifrele nenule, n , și se cere să se scrie numărul obținut prin eliminarea tuturor cifrelor impare din n , respectiv -1 dacă nu există astfel de cifre sau dacă toate cifrele lui n sunt impare. (Examen de Bacalaureat - Iunie 2016)
7. Se citește un număr natural cu toate cifrele nenule, n . Să se scrie numărul obținut prin eliminarea tuturor cifrelor pare din n , respectiv -1 dacă nu există astfel de cifre sau dacă toate cifrele lui n sunt pare.
8. Se citește un număr natural n . Afișați numerele care se pot obține din n , prin eliminarea unei singure cifre.
9. Se citește un număr natural n . Afișați cel mai mare număr care se poate obține din n prin eliminarea unei singure cifre.
10. Se citește un număr natural n cu cel puțin 3 cifre. Să se calculeze și să se afișeze numărul obținut din n prin eliminarea primei și ultimei cifre.
11. Se citesc de la tastatură două numere naturale n și k . Să se elimine k cifre din numărul n , astfel încât numărul rămas să fie maxim.
12. Se dă un număr natural n . Să se afle câte dintre numerele obținute din n prin ștergerea unei cifre, sunt divizibile cu 3.
13. Un număr natural n se împarte la toate numerele obținute din el prin eliminarea unei cifre. Să se afișeze resturile care se obțin.
14. Se dau n numere naturale. Calculați suma numerelor obținute prin eliminarea din fiecare număr dat a ultimei cifre.
15. Se dau n numere naturale. Afișați numerele obținute prin eliminarea din fiecare număr dat a primei cifre.
16. Se dau n numere naturale. Calculați suma numerelor obținute prin eliminarea din fiecare număr dat a tuturor aparițiilor ultimei cifre.
17. Se dau n numere naturale. Calculați suma numerelor obținute prin eliminarea din fiecare număr dat a tuturor aparițiilor primei cifre.
18. Fiind date două numere a și b , îl numim pe a *suffix* al lui b dacă a este egal cu b sau dacă b se poate obține din a prin alipirea la stânga a unor noi cifre.
19. Se dă un număr natural x și un șir de numere naturale. Să se determine numerele din șir care îl au ca suffix pe x .
20. Se citește un număr natural n . Afișați numărul obținut astfel: prima cifră inversată cu a 2-a, a 3-a cu a 4-a etc.
21. Se citește de la tastatură un număr natural n și o cifră c . Să se adauge la sfârșitul numărului n cifra c și să se afișeze numărul astfel obținut.
22. Se citește de la tastatură un număr natural n și o cifră c . Să se adauge pe prima poziție a numărului n cifra c și să se afișeze numărul astfel obținut.

23. Se citește de la tastatură un număr natural n și o cifră c . Să se adauge pe poziția k a numărului n cifra c și să se afișeze numărul astfel obținut (cifra unităților se află pe poziția 1).
24. Se citește un număr natural n . Afișați cel mai mare număr care se poate obține din n prin adăugarea unei cifre c date.
25. Se dă un număr natural n . Să se afle câte dintre numerele obținute din n prin adăugarea unei cifre c , sunt divizibile cu 3.
26. Se dă un număr natural n . Să se adauge pe pozițiile pare din numărul inițial o cifră c dată (cifra unităților se află pe poziția 1).
27. Se dau n numere naturale. Calculați suma numerelor obținute prin adăugarea unei cifre c pe prima poziție pentru fiecare dintre cele n numere citite (se consideră că cifra unităților se află pe ultima poziție).
28. Se dau n numere naturale. Calculați suma numerelor obținute prin adăugarea unei cifre c pe ultima poziție pentru fiecare dintre cele n numere citite (se consideră că cifra unităților se află pe ultima poziție).
29. Se citește un număr natural, n ($n \geq 1$), și se cere să se scrie numărul obținut prin duplicarea fiecărei cifre pare a lui n sau -1 dacă acesta nu are nici o cifră pară. (Examen de Bacalaureat - August 2016)
30. Se citește un număr natural nenul, n , și se cere să se scrie numărul obținut din n , prin inserarea, după fiecare cifră pară din scrierea lui n , a unei cifre egale cu jumătate din aceasta. (Examen de Bacalaureat - Iunie 2017)
31. Scrieți un program care înlocuiește în numărul n toate aparițiile cifrei c_1 cu c_2 .
32. Se citește de la tastatură un număr natural n . Să se înlocuiască cifra de pe ultima poziție din număr cu o cifră c dată (se consideră că cifra unităților se află pe ultima poziție).
33. Se citește de la tastatură un număr natural n . Să se înlocuiască cifra de pe prima poziție din număr cu o cifră c dată (se consideră că cifra unităților se află pe ultima poziție).
34. Se citește de la tastatură un număr natural n , care conține un număr impar de cifre. Să se înlocuiască cifra din mijloc cu o cifră c dată.
35. Se citește de la tastatură un număr natural n . Să se înlocuiască cifrele de pe pozițiile pare din număr cu o cifră c dată.
36. Să se înlocuiască cifrele pare dintr-un număr cu o cifră c dată.
37. Se dau n numere naturale. Calculați suma numerelor obținute prin înlocuirea primei cifre din fiecare număr cu o cifră c dată (se consideră că cifra unităților se află pe ultima poziție).
38. Se citește un număr natural n cu cel mult 9 cifre. Afișați numerele care se pot construi din n prin permutarea cifrelor sale succesiv cu câte o poziție spre dreapta.

39. Se citește un număr natural n cu cel mult 9 cifre. Afișați numerele care se pot construi din n prin permutarea cifrelor sale succesiv cu câte o poziție spre stânga.
40. Se citește un număr natural n cu cel mult 9 cifre. Afișați numerele care se pot construi din n prin permutarea cifrelor sale succesiv cu câte k poziții spre dreapta (k este dat, k este mai mic decât numărul de cifre ale lui n).
41. Se citește un număr natural n cu cel mult 9 cifre. Afișați numerele care se pot construi din n prin permutarea cifrelor sale succesiv cu câte k poziții spre stânga (k este dat, k este mai mic decât numărul de cifre ale lui n).

2.6. Baze de numerație

1. Să se transforme un număr din baza 10 în baza 2.
2. Să se transforme un număr din baza 10 în baza 16.
3. Să se transforme un număr din baza 10 într-o bază b ($b < 10$).
4. Să se transforme un număr din baza b în baza 10.
5. Se dă un număr scris în baza 2. Să se afișeze valoarea acestuia în baza 4.
6. Să se transforme un număr din baza b în baza q ($b, q < 10$).
7. Se citește un număr natural n . Să se determine câte cifre 0 și câte cifre 1 are reprezentarea în baza 2 a acestui număr.
8. Se citesc două numere naturale n_1 scris în baza b_1 și n_2 scris în baza b_2 . Să se afișeze maximul dintre n_1 și n_2 (transformate în baza 10).
9. Se citesc două numere naturale. Să se afișeze numărul care are cele mai multe cifre egale cu 1 în reprezentarea în baza 2.
10. Se consideră un număr natural n și o bază de numerație b . Să se determine cea mai mare cifră a numărului n transformat în baza b .
11. Se citește de la tastatură un număr natural n și o bază b . Să se verifice dacă n poate fi scrierea în baza b a unui număr n .
12. Se dau n numere naturale. Determinați cea mai mică bază de numerație b în care ar putea fi considerate aceste numere.
13. Se citesc de la tastatură două numere naturale a și b . Să se transforme aceste numere în baza 2, apoi să se afișeze pozițiile care conțin aceleași valori.
14. Se citește de la tastatură un număr natural n în baza 10. Să se verifice dacă numărul poate fi palindrom într-o bază b dată.
15. Să se verifice dacă un număr n dat este palindrom în baza 16.
16. Se citește un număr b , reprezentând o bază de numerație și apoi n cifre, reprezentând cifrele unui număr scris în baza b . Determinați transformarea acestui număr în baza 10.

17. Fiind date n numere naturale, aflați câte dintre acestea se pot scrie ca sumă de puteri distincte ale lui 2.
18. Un număr natural în baza 10 se numește *prețios* dacă numărul de cifre ale sale din baza 2 este număr prim. Să se verifice dacă un număr n dat este număr *prețios*.
19. Se consideră un interval $[a, b]$. Determinați câte numere *prețioase* se află în acest interval.

2.7. Cel mai mare divizor comun. Cel mai mic multiplu comun

1. Să se scrie un program care să determine *cel mai mare divizor comun* a două numere naturale citite de la tastatură.
2. Să se scrie un program care să determine *cel mai mic multiplu comun* a două numere naturale citite de la tastatură.
3. Se citesc două numere naturale a și b . Să se verifice dacă sunt prime între ele (două numere sunt prime între ele dacă cel mai mare divizor comun al lor este 1).
4. Să se scrie un program care citește de la tastatură două numere naturale a și b , reprezentând numărătorul, respectiv numitorul unei fracții și afișează fracția ireductibilă, obținută din fracția inițială.
5. Se citesc de la tastatură 2 perechi de numere, reprezentând numărătorii și numitorii a 2 fracții. Să se calculeze suma celor două fracții și să se afișeze fracția sumă în formă ireductibilă.
6. Se citește un număr natural n . Acest număr se *împarte* în alte două numere a și b , astfel: a este format din cifrele din prima jumătate a lui n , b este format din cifrele din a doua jumătate a lui n . Dacă n are număr impar de cifre, cifra din mijloc se ignoră. Să se determine *cmmdc* a celor 2 numere astfel formate.
7. Se citește de la tastatură un număr natural n . Să se afișeze mulțimea numerelor naturale mai mici decât n care sunt prime cu n .
8. Se citesc două numere naturale a și b . Afișați toate perechile de numere x și y din intervalul $[a, b]$ care au proprietatea că sunt prime între ele.
9. Se citesc două numere naturale a și b . Afișați toate perechile de numere x și y din intervalul $[a, b]$ care au proprietatea că sunt prime între ele, dar x și y nu sunt numere prime.
10. Se citesc numere naturale de la tastatură până la apariția lui 0. Să se determine câte perechi de numere citite consecutiv sunt prime între ele.
11. Se dau n numere naturale. Determinați câte dintre ele au proprietatea că sunt prime cu oglinditul lor.
12. Să se calculeze *cmmdc* a 3 numere a , b și c date.

13. Să se scrie un program care citește un șir de n numere naturale și determină cea mai mică fracție care poate fi scrisă cu numărătorul și numitorul (numere consecutive) dintre cele n numere. Se va afișa fracția în forma ireductibilă.
14. Să se scrie un program care calculează câte perechi de numere naturale mai mici decât n (n este dat) au *cmmdc* un număr d dat.
15. Se citește de la tastatură un număr natural nenul n . Să se determine și să se afișeze *indicatorul lui Euler* pentru toate numerele naturale mai mici sau egale cu n . Indicatorul lui Euler pentru un număr n dat este egal cu numărul numerelor prime cu n , mai mici decât n .
16. Se dau n numere naturale nenule. Calculați *cel mai mare divizor comun* al lor.
17. Se dau n numere naturale nenule. Calculați *cel mai mic multiplu comun* al lor.
18. Se dau n perechi de numere naturale nenule, fiecare reprezentând câte o fracție. Calculați suma celor n fracții, aducând rezultatul la forma ireductibilă.
19. Se citesc perechi de numere naturale până la citirea unei valori nule. Să se calculeze suma numerelor din perechile pentru care atât valorile inițiale cât și oglinditele lor sunt prime între ele.
20. Dându-se șirul de fracții $1/n, 2/n, 3/n, \dots, n/n$, să se afle câte fracții sunt ireductibile. Numărul n este dat de la tastatură.
21. Avem la dispoziție un chenar dreptunghiular format din oglinzi. O rază de lumină pornește din colțul stânga jos al dreptunghiului sub un unghi de 45 de grade față de latura de jos a dreptunghiului și lovește latura de sus sau latura din dreapta. Aici se reflectă (pornește spre o altă latură tot sub un unghi de 45 de grade față de latura de care s-a lovit). Își continuă drumul până când ajunge într-un colț al dreptunghiului. (Admitere UBB-Cluj 2017)

2.8. Puterile lui n

1. Să se calculeze n la puterea p fără a folosi o funcție predefinită.
2. Se citesc două numere naturale n și m . Afișați în ordine crescătoare toate puterile lui n care sunt mai mici sau egale decât m (Pentru $n = 2$ și $m = 17$ se vor afișa 2, 4, 8, 16).
3. Se citește de la tastatură un număr natural p . Să se scrie primele n puteri ale numărului p ($p, p^2, p^3, \dots, p^n$).
4. Să se determine și să afișeze primul și ultimul număr de k cifre (k este un număr natural dat).
5. Se citesc trei numere naturale n , a și b . Afișați puterile lui n din intervalul $[a, b]$.

6. Să se verifice dacă un număr n se poate scrie ca sumă de puteri ale lui k (k este dat de la tastatură).
7. Să se verifice dacă un număr n se poate scrie ca sumă de puteri distincte ale lui k (k este dat).
8. Se citesc de la tastatură n numere naturale. Să se precizeze câte dintre aceste numere se pot scrie ca sumă de puteri distincte ale lui k (k este dat).
9. Să se calculeze suma: $1 + 2 + 2^2 + 2^3 + \dots + 2^n$, pentru un număr natural n , dat de la tastatură.
10. Să se calculeze suma: $1 + k + k^2 + k^3 + \dots + k^n$, pentru n, k date de la tastatură.
11. Să se calculeze suma: $1 + 2^2 + 3^3 + \dots + n^n$, pentru un număr natural n , dat de la tastatură.

2.9. Generări de șiruri

1. Să se afișeze primii n termeni ai șirului 1,2,4,8,... Numărul natural n este citit de la tastatură.
2. Să se afișeze primele n numere din șirul 1, 2,2, 3,3,3, 4,4,4,4,..., unde n este un număr citit de la tastatură.
3. Să se precizeze valoarea elementului de pe poziția k dată, a șirului 1, 2,2, 3,3,3, 4,4,4,4....
4. Să se genereze primele n numere din șirul 1 1,2,2 1,2,3,3,3 1,2,3,4,4,4,4 Numărul natural n este citit de la tastatură.
5. Să se scrie un program eficient din punct de vedere al timpului de executare și al spațiului de memorie, care citește de la tastatură un număr natural n ($n < 1000$) și afișează pe ecran cel de-al n -lea termen al șirului 1, 1,2,2, 1,2,3,3,3, 1,2,3,4,4,4,4 ..., construit astfel: prima grupă este formată din numărul 1, a doua grupă este formată din numărul 1, urmat de numărul 2 scris de două ori etc. Grupa a k -a va fi formată din numerele 1,2,3,..., $k-1$ urmate de numărul k scris de k ori. (Variante Bacalaureat 2009)
6. Să se afișeze primele n numere din șirul 3, 6, 12, 24, 48, ..., unde n este un număr citit de la tastatură, iar fiecare dintre termeni se obține din precedentul înmulțit cu 2.
7. Să se afișeze primele n numere din șirul 1, 3, 7, 15, 31, ..., unde n este un număr citit de la tastatură, iar fiecare termen i este de forma $2(i-1) + 1$.
8. Să se afișeze primii n termeni ai șirului 1,2,1 1,2,3,2,1 1,2,3,4,3,2,1,..., unde n este un număr citit de la tastatură.
9. Să se genereze primii n termeni din șirul 1, 2,1, 3,2,1, 4,3,2,1, 5,4,3,2,1...

10. Să se genereze primii n termeni din șirul 1, 2,1, 3,2,1, 4,2,2, 5,4,3,2,1, 6,2,2,3,3,3, 7,6,5,4 ...
11. Fie n un număr natural nenul. Să se genereze primii n termeni consecutivi ai șirului 1, 2,2, 1,2,3, 4,4,4,4, 1,2,3,4,5, 6,6, ...
12. Să se genereze primii n termeni ai șirului 1, 1,2,4, 3,9,27, 1,4,16, 5,25,125... unde n este un număr natural dat de la tastatură.
13. Se consideră șirul crescător format din toate numerele naturale nenule, în care fiecare număr apare de exact k ori. De exemplu, pentru $k = 3$ termenii șirului sunt 1,1,1, 2,2,2, 3,3,3, 4,4,4, 5,5,5,... Scrieți un program care citește două numere naturale k și n și afișează pe ecran primii n termeni ai șirului, separați prin câte un spațiu.
14. Se consideră șirul crescător format din toate numerele naturale nenule, în care fiecare număr apare de exact k ori. De exemplu, pentru $k = 3$ termenii șirului sunt 1,1,1, 2,2,2, 3,3,3, 4,4,4, 5,5,5,... Scrieți un program care citește două numere naturale k și n și afișează pe ecran primii n termeni ai șirului, separați prin câte un spațiu, în ordine descrescătoare. (Variante Bacalaureat 2009)
15. Se consideră următorul șir de numere naturale: 1, 2,2, 3,3,3, 4,4,4,4, 5,5,5,5,5... Primul element este 1 și se află pe poziția 1 în șir. Un număr x apare de x ori în șir pe poziții consecutive și elementele șirului sunt în ordine crescătoare. Pentru o valoare naturală n ($0 < n < 10000$), citită de la tastatură se cere să se precizeze care este termenul cu numărul de ordine n din șirul prezentat. (Variante Bacalaureat 2009)
16. Se consideră următorul șir de numere naturale: 1, 3, 5, 8, 10, 14, 16..., în care termenul de pe poziția i ($i > 1$) este egal cu cel precedent plus numărul de divizori ai lui i . Afișați al n -lea termen.
17. Se consideră șirul 1, 2,1, 3,2,1, 4,3,2,1, ..., în care prima grupă este formată din numărul 1, a doua grupă este formată din numerele 2,1, ..., a k -a grupă este formată din numerele $k, k-1, \dots, 2, 1$. Pentru un număr natural dat n , să se determine al n -lea termen din șir. (Variante Bacalaureat 2009)
18. Se citește de la tastatură un număr natural nenul n . Să se genereze și să se afișeze primii n termeni ai șirului:

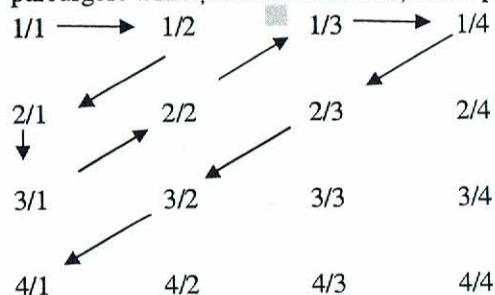
$$\frac{1}{1}, \frac{2}{1}, \frac{1}{2}, \frac{3}{1}, \frac{2}{2}, \frac{1}{3}, \frac{4}{1}, \frac{3}{2}, \frac{2}{3}, \frac{1}{4}, \dots$$

19. Se consideră șirul definit mai jos:

$$f_n = \begin{cases} 1, & \text{dacă } n = 1 \\ 1 + f_{n-1}, & \text{dacă } n \text{ este par} \\ 1 + 2 \cdot f_{n-2}, & \text{altfel} \end{cases}$$

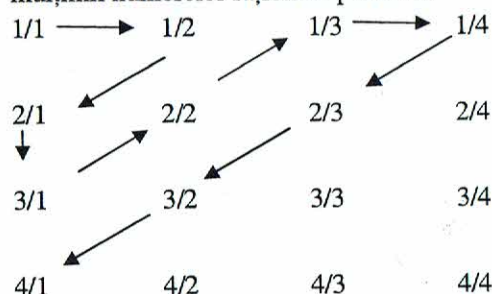
Să se afișeze primii n termeni ai șirului definit mai sus (n este un număr natural dat de la tastatură).

20. Șirul lui Cantor. Se consideră următoarea modalitate de reprezentare și parcurgere a mulțimii numerelor raționale pozitive:



Fiind dat un număr natural n , să se determine cel de-al n -lea termen din șir.

21. Se consideră următoarea modalitate de reprezentare și parcurgere a mulțimii numerelor raționale pozitive:



Fiind dat un element de forma x/y , să se determine al câtelea termen este el în cadrul șirului lui Cantor.

22. Să se genereze următorul triunghi, unde n este un număr natural dat de la tastatură.

```
1
1 2
1 2 3
.....
```

```
1 2 3 .... n.
```

23. Să se genereze următorul triunghi, unde n este un număr natural dat de la tastatură.

```
n n-1 n-2 .... 3 2 1
...
3 2 1
2 1
1
```

24. Să se genereze următorul triunghi, unde n este un număr natural dat de la tastatură.

```
2
4 6
8 10 12
14 16 18 20
.....
```

Să se afișeze primele k linii ale triunghiului definit mai sus.

2.10. Șirul lui Fibonacci

1. Șirul lui Fibonacci se definește astfel: $f_1 = 1, f_2 = 1$ și $f_n = f_{n-1} + f_{n-2}$. Să se afișeze termenii din șirul lui Fibonacci mai mici decât o valoare dată n .
2. Să se afișeze primii n termeni din șirul lui Fibonacci. Valoarea lui n este citită de la tastatură.
3. Se citește de la tastatură un număr natural x . Să se verifice dacă acest număr este termen în șirul lui Fibonacci.
4. Se citesc de la tastatură două numere naturale nenule a și b . Să se verifice dacă cele două numere sunt doi termeni consecutivi ai șirului lui Fibonacci.
5. Se citește un număr natural n . Să se afișeze cea mai scurtă descompunere a lui n ca sumă de termeni distincți din șirul lui Fibonacci.
6. Se citesc două numere a și b . Calculați și afișați câți termeni din șirul lui Fibonacci se află în intervalul $[a, b]$.
7. Se dau n numere naturale. Să se precizeze câte numere din cele citite sunt termeni ai șirului lui Fibonacci.
8. Se citește un număr natural nenul, n și se cere să se scrie valoarea celui de al n -lea termen impar al șirului lui Fibonacci. (Examen de Bacalaureat - Iunie 2015)

2.11. Maxime și minime

1. Se dau n numere întregi. Afișați cel mai mare dintre cele n numere date.
2. Se dau n numere întregi. Afișați cel mai mic dintre cele n numere date.
3. Se consideră n numere naturale. Calculați suma dintre cel mai mare și cel mai mic număr dat.
4. Se citesc numere de la tastatură până la apariția lui 0. Să se determine maximul lor.

5. Se citesc numere de la tastatură până la apariția lui 0. Să se determine minimul valorilor pozitive.
6. Se citește un număr natural k și un număr natural n . Se citesc apoi n numere naturale. Să se afișeze cel mai mare dintre cele n numere citite, care au exact k divizori.
7. Se citesc de la tastatură n numere naturale. Afișați cel mai mic număr palindrom dintre cele citite. În cazul în care șirul nu conține nici un număr palindrom, se va afișa un mesaj corespunzător.
8. Se citește un număr natural n și apoi n numere naturale cu maxim 4 cifre fiecare. Calculați și afișați numărul obținut prin lipirea la numărul maxim, a numărului minim dintre cele n numere citite.
9. Se citesc n numere naturale. Să se afișeze cel mai mare număr prim dintre cele citite.
10. Se dau n numere naturale. Se construiește un număr selectând de la fiecare număr cifra maximă. Afișați acest număr.
11. Se citește un număr natural de maxim 9 cifre. Să se afișeze cea mai mare și cea mai mică cifră din număr.
12. Să se scrie un program care citește un șir de n numere naturale și determină numărul din șir care are prima cifră minimă.
13. Să se scrie un program care citește un șir de n numere naturale și determină cele mai mari două numere din șir.
14. Să se scrie un program care citește un șir de n numere naturale și determină perechea de valori consecutive din șir pentru care modulul diferenței este minim.
15. Se consideră un șir cu n elemente numere reale, numerotate de la 1 la n . Determinați numărul de ordine al valorii maxime din șir.
16. Se citesc de la tastatură n numere naturale. Să se determine numărul a cărui sumă a cifrelor este cea mai mare (respectiv cea mai mică).
17. Se citesc două numere naturale a și b . Să se determine cel mai mic și cel mai mare număr din intervalul $[a, b]$ cu număr maxim de divizori.
18. Se consideră un număr natural n . Să se determine numărul din intervalul $[1, n]$ care are număr maxim de divizori. În cazul în care există mai multe astfel de numere se vor afișa toate.
19. Se citesc de la tastatură șiruri nevide de numere întregi, fiecare terminat cu 0, iar ultimul șir citit terminat cu -1. Să se determine minimul fiecărui șir și minimul global (al tuturor șirurilor).

2.12. Probleme olimpiadă

Problema 1 - Palindrom

Un număr se numește *palindrom* dacă prima lui cifră este egală cu ultima, a doua cu penultima și așa mai departe. De exemplu numerele 1221, 505 și 7 sunt palindromuri, în vreme ce 500, 1410 și 2424 nu sunt palindromuri.

Similar, un număr se numește *aproape palindrom* dacă are aceleași perechi de cifre identice ca un palindrom, mai puțin o pereche în care cifrele diferă. De exemplu numerele 500, 1411, 2444, 1220, 53625, 14 și 4014 sunt numere aproape palindromuri (cu perechea de cifre neidentice îngroșată), în vreme ce 1221, 1410, 6, 505, 22 și 512125 nu sunt numere aproape palindromuri deoarece fie sunt palindromuri, fie au prea multe perechi de cifre diferite.

Mai definim *palindromul asociat* al unui număr x ca fiind cel mai mic număr palindrom p strict mai mare decât x ($p > x$). De exemplu palindromul asociat al lui 5442 este 5445, palindromul asociat al lui 2445 este 2552, al lui 545 este 555, al lui 39995 este 40004, al lui 500 este 505, iar al lui 512125 este 512215.

Cerințe

Scrieți un program care citind un număr natural nenul n și apoi un șir de n numere naturale determină:

1. câte dintre cele n numere sunt palindrom
2. câte dintre cele n numere sunt aproape palindrom
3. palindromurile asociate pentru cele n numere citite.

(Olimpiada județeană, 2016)

Problema 2 - Echilibrat

Numim număr *echilibrat* un număr natural pentru care suma cifrelor de pe poziții pare este egală cu suma cifrelor de pe poziții impare.

De exemplu numărul 13552 este echilibrat, pentru că $1+5+2=8=3+5$.

Cerință

Dat fiind un număr natural N să se determine cel mai mic număr echilibrat, strict mai mare decât N .

(Olimpiada județeană, 2015)

Problema 3 - Minim

Avem la dispoziție patru numere naturale N, A, B, C , precum și trei cifre c_1, c_2, c_3 distincte două câte două.

Cerință

Să se determine numărul natural minim, strict mai mare decât N , care are exact A cifre c_1 , B cifre c_2 , C cifre c_3 și nu conține alte cifre.

(Olimpiada județeană, 2014)

Problema 4 - Generare

Se generează un șir de numere naturale ai cărui primi termeni sunt, în ordine: 1, 12, 21, 123, 231, 312, 1234, 2341, 3412, 4123, 12345, 23451,...

Cerințe:

Deduceți regula după care sunt generați termenii șirului și scrieți un program care să citească numerele naturale k, x, a și b și care să determine:

- ultima cifră a sumei tuturor termenilor șirului care sunt formați din cel mult k cifre;
- succesorul termenului x în șirul dat, x fiind un termen al șirului;
- numărul de termeni ai șirului care au cifra cea mai semnificativă egală cu a și nu conțin în scrierea lor cifra b .

(Olimpiada județeană, 2010)

3. TIPURI STRUCTURATE DE DATE

3.1. Tablouri unidimensionale (Vectori)

3.1.1. Parcurgeri

1. Fiind dat un vector de numere întregi, să se determine suma elementelor sale.
2. Fie v un vector de numere întregi. Să se determine suma elementelor pozitive.
3. Să se afișeze elementele pare dintr-un vector v cu n elemente numere întregi.
4. Scrieți un program care afișează elementele pare aflate pe poziții impare dintr-un vector de numere întregi.
5. Să se afișeze elementele pozitive de rang impar dintr-un vector de numere întregi.
6. Să se scrie un program care afișează numărul elementelor pozitive, negative și respectiv nule dintr-un vector de numere întregi.
7. Fie v un vector cu n elemente numere întregi. Să se precizeze de câte ori apare o valoare dată x în vector.
8. Să se precizeze de câte ori se găsește o valoare dată x printre elementele unui vector v cu n elemente numere întregi, sortat crescător.
9. Să se calculeze media aritmetică a elementelor impare dintr-un vector v cu n elemente numere întregi.
10. Fiind dat un vector cu n elemente întregi, scrieți un program care calculează suma și numărul elementelor pozitive aflate pe poziții impare din vector.

11. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se determine ultima cifră a sumei elementelor din vector.
12. Se consideră un șir cu n elemente numere naturale. Determinați diferența în valoare absolută dintre numărul de valori pare și numărul de valori impare din șir.
13. Se consideră un vector v cu n componente numere întregi. Să se determine câte elemente din vector sunt strict mai mici decât ma și câte sunt strict mai mari decât ma , unde ma este media aritmetică a componentelor din vector.
14. Se consideră un vector v cu n componente numere întregi. Să se determine câte elemente din vector au același semn cu ma , unde ma este media aritmetică a componentelor din vector.
15. Se citește un vector v cu n elemente numere întregi. Afișați numărul de perechi de elemente din vector care au suma divizibilă cu n .
16. Se citește un vector cu n elemente, numere naturale. Să se afișeze elementele din vector care sunt divizori ai ultimului element.
17. Se citește un vector v cu n elemente numere întregi. Afișați numărul de perechi egal depărtate de capetele vectorului care au proprietatea că încep cu aceeași cifră.
18. Se citește un vector v cu n elemente numere întregi. Parcurgeți și afișați elementele simetrice primul, ultimul, al doilea, penultimul etc.
19. Se citește un vector v cu n elemente numere naturale. Afișați perechile de elemente din vector care au proprietatea că sunt prime între ele.
20. Se citește un vector v cu n elemente numere naturale. Afișați perechile consecutive din vector care au proprietatea că sunt prime între ele.
21. Se citește de la tastatură un șir de n numere naturale. Să se determine elementul din șir, cel mai apropiat de un pătrat perfect.
22. Să se afișeze elementele numere prime ale unui vector v cu n elemente numere naturale.
23. Să se afișeze elementele dintr-un vector v care sunt prime și care citite invers sunt tot numere prime.
24. Precizați câte dintre elementele unui vector v cu n elemente numere întregi au exact k divizori proprii (k este dat).
25. Să se calculeze $cmmdc$ al elementelor unui vector v cu n elemente numere întregi.
26. Să se precizeze câte dintre elementele unui vector v cu n elemente sunt prime cu un număr n dat.
27. Se dă un vector cu n numere naturale. Să se determine câte dintre perechile de elemente din vector sunt prime între ele.

28. Să se afișeze elementele numere perfecte dintr-un vector v cu n componente numere naturale (Un număr este *perfect*, dacă este egal cu suma divizorilor săi fără el însuși).
29. Să se precizeze cu câte zerouri se termină produsul elementelor unui vector v cu n numere naturale.
30. Fie v un vector cu n elemente numere întregi. Să se afișeze toate perechile de elemente ale vectorului cu proprietatea că ambele elemente ale perechii au același număr de cifre.
31. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se determine câte dintre elementele vectorului se află în intervalul $[a, b]$ dat.
32. Se citește un vector cu n elemente, numere naturale. Să se afișeze elementele pare în ordinea crescătoare a indicilor, iar elementele impare în ordinea descrescătoare a indicilor.
33. Se dă un vector cu n numere naturale. Să se determine câte dintre perechile de elemente simetrice (egal depărtate de mijlocul vectorului) sunt prime între ele.
34. Să se afișeze și să se numere câte din elementele unui vector dat de numere naturale sunt *numere Fibonacci*.
35. Se citesc de la tastatură doi vectori v și u cu elemente numere întregi de dimensiuni n , respectiv m . Să se numere câte elemente din al doilea vector sunt strict mai mici decât toate elementele din primul vector. Dacă nu există nici un element cu proprietatea dată se va afișa mesajul *Nu există*.
36. Să se calculeze suma a două *numere mari*. (Un număr mare este un număr care nu se poate citi într-o variabilă predefinită. Pentru citirea lui se va folosi un vector).
37. Să se calculeze diferența a două *numere mari*. (Un număr mare este un număr care nu se poate citi într-o variabilă predefinită. Pentru citirea lui se va folosi un vector).
38. Să se calculeze 2^n pentru valori mari ale lui n .
39. Să se determine numărul de cifre egale cu c (c dat de la tastatură) în cadrul numărului $n!$ ($n < 400$).
40. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se verifice dacă elementele sunt în progresie aritmetică și să se afișeze un mesaj corespunzător.
41. Se citește de la tastatură un număr natural n , apoi n numere întregi. Să se verifice dacă vectorul conține un număr *majoritar* (Un număr este *majoritar* dacă numărul său de apariții este mai mare decât $n/2$).
42. Se consideră două mulțimi A și B , date prin intermediul a doi vectori. Să se determine intersecția, reuniunea și diferența celor două mulțimi.
43. Să se descompună în factori primi un număr natural x dat. Pentru memorarea rezultatelor se vor folosi doi vectori. Un vector v care să memoreze factorii primi și un vector u care să memoreze puterile factorilor primi. Să se afișeze cei doi vectori.

44. Se consideră doi vectori u și v cu câte n elemente fiecare. Să se calculeze media aritmetică a elementelor din u , pentru care elementele respective din v sunt numere prime.
45. *Relațiile lui Viète*. Fie x un vector cu n elemente numere întregi. Să se calculeze $s_1, s_2, \dots, s_n$, unde

$$s_1 = x_1 + x_2 + \dots + x_n$$

$$s_2 = x_1 \cdot x_2 + x_1 \cdot x_3 + \dots + x_{n-1} \cdot x_n$$

$$\dots$$

$$s_n = x_1 \cdot x_2 \cdot x_3 \cdot \dots \cdot x_n.$$
 Sumele de mai sus sunt date de relațiile lui Viète pentru polinomul $P = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$.

3.1.2. Construcții de vectori

1. Se consideră un vector v cu n elemente numere întregi. Să se copieze într-un vector u elementele pozitive ale vectorului inițial.
2. Se consideră un vector v cu n elemente numere întregi. Să se copieze într-un vector u elementele pare ale vectorului inițial.
3. Se consideră un vector v cu n elemente numere întregi și două numere a și b . Să se copieze într-un vector u elementele din v aflate în intervalul $[a, b]$.
4. Se consideră un vector v cu n elemente numere întregi și două numere a și b . Să se copieze într-un vector u elementele din v aflate în intervalul $[a, b]$, în ordinea inversă a apariției lor în vector.
5. Se consideră doi vectori u și v cu n și respectiv m elemente numere întregi. Se citește un număr natural k . Să se interclaseze în primul vector, după poziția k , tabloul al doilea.
6. Fie v un vector de numere întregi. Să se construiască un vector u astfel încât u_i să conțină numărul de apariții ale lui v_i în vectorul v .
7. Fie v un vector cu n elemente numere întregi. Să se construiască alți doi vectori care să conțină elementele pare, respectiv impare ale vectorului inițial.
8. Se dă un vector x cu n elemente, numere naturale. Să se construiască un vector y , care să conțină elementele prime din x .
9. Se dă un vector x cu n elemente, numere naturale. Să se construiască un vector y , care să conțină elementele prime din x , în ordine inversă.
10. Se consideră un vector v cu n elemente numere naturale. Să se construiască un vector u cu proprietatea că u_i este egal cu restul împărțirii lui v_i la suma cifrelor lui v_i .
11. Un vector de șiruri de caractere conține elemente care pot avea una din două valori: "alb" sau "negru". Să se construiască un vector care conține mai întâi indecșii elementelor "alb" și apoi indecșii elementelor "negru".

12. Se citește de la tastatură un șir de n numere naturale. Să se împartă numerele din șir în grupe (elementele fiecărei grupe vor fi memorate într-un vector), astfel încât toate elementele aceleși grupe să aibă aceeași cifră de control. Să se afișeze grupele astfel obținute, câte una pe fiecare rând. (pentru un număr *cifra de control* se obține astfel: se calculează suma cifrelor lui x , apoi suma cifrelor sumei obținute, apoi suma cifrelor noii sume ș.a.m.d. până când ajungem la o sumă alcătuită dintr-o singură cifră.)
13. Se citește de la tastatură un șir cu n numere naturale. Să se afișeze toate grupurile de numere din șir care au aceeași divizori primi. Fiecare grup va fi memorat într-un vector și se va afișa pe câte un rând.
14. Se citește de la tastatură un număr n , apoi n numere întregi, elemente ale unui vector v . Să se construiască un vector u , astfel încât u_i să conțină numărul de valori din v pentru care $v_j < u_i$ ($1 \leq j \leq i - 1$).

3.1.3. Operații cu vectori

1. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze pe ultima poziție a vectorului un element x .
2. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze pe prima poziție a vectorului un element x .
3. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze pe poziția k a vectorului un element x (k este dat de la tastatură).
4. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine elementul de pe ultima poziție din vector.
5. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine elementul din vector de pe prima poziție.
6. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine elementul de pe poziția k din vector (k este dat de la tastatură).
7. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector elementul maxim (vectorul conține un singur element maxim).
8. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector elementul cu suma cifrelor minimă (vectorul conține un singur element care îndeplinește cerința).
9. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector elementul care are prima cifră egală cu 2 (vectorul conține un singur element cu prima cifră egală cu 2).
10. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector elementul care este număr prim (vectorul conține un singur element număr prim).
11. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector toate elementele numere negative.

12. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector toate elementele impare.
13. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector elementul maxim (vectorul poate conține mai multe elemente egale cu valoarea maximă).
14. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector numerele palindrom.
15. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector toate elementele care au prima cifră număr par.
16. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector toate elementele care au suma cifrelor divizibilă cu un număr p dat.
17. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector numerele pătrate perfecte.
18. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se elimine din vector elementele aflate pe poziții divizibile cu 3.
19. Se citește un vector v cu n elemente numere întregi. Să se elimine din toate elementele vectorului cifrele impare și să se afișeze vectorul astfel obținut.
20. Se citește de la tastatură un vector v cu n elemente numere naturale. Să se elimine în mod repetat toate perechile de numere alăturate identice.
21. Se citește un vector v cu n elemente numere întregi. Să se elimine un număr minim de elemente din vectorul v astfel încât elementele rămase să fie ordonate strict crescător. Primul element din vector nu se elimină.
22. Se citește un vector v cu n elemente numere întregi. Să se șteargă toate aparițiile primului element și să se afișeze vectorul rămas.
23. Fie v un vector cu n elemente numere întregi. Să se scrie un program care elimină toate elemente nule din vector.
24. Se consideră un vector v cu n elemente numere întregi. Să se scrie un program care elimină din vector valorile ce se află în intervalul $[a, b]$, unde a și b sunt date.
25. Numerele de la 1 la n sunt așezate pe circumferința unui cerc în ordine crescătoare, astfel încât n să ajungă lângă 1. Începând cu numărul x se elimină numerele din k în k , după fiecare eliminare cercul modificându-se. Să se precizeze ultimul număr rămas.
26. Să se verifice dacă un vector dat este o mulțime (toate elementele vectorului sunt distincte). Dacă nu este mulțime, să se transforme acest vector în mulțime.
27. Se consideră un șir de n numere întregi, cu n număr natural nenul. Se elimină primul element din șir și toate elementele șirului aflate pe poziții care reprezintă numere prime, în ordinea crescătoare a pozițiilor. Operația de eliminare se repetă cu elementele rămase în șir, repositionate după eliminarea celorlalte, până când este eliminat și ultimul element rămas.

Să se scrie un program care a fișează elementele șirului inițial, în ordinea în care au fost eliminate conform algoritmului descris mai sus.

(Admitere FMI București – 2016)

28. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se înlocuiască cu 0 toate elementele din vector mai mici sau egale decât prima valoare din tablou.
29. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se înlocuiască cu -1 toate elementele din vector care sunt numere prime.
30. Se dă un vector v cu n elemente numere întregi. Fără a folosi un vector auxiliar, să se mute la sfârșitul lui v elementele sale nule, păstrând ordinea celorlalte elemente.
31. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se mute pe ultimele poziții elementele impare.
32. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se mute pe ultimele poziții elementele cu suma cifrelor un număr par.
33. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se mute pe ultimele poziții elementele pătrate perfecte.
34. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se mute pe ultimele poziții elementele numere prime.
35. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se mute pe ultimele poziții elementele care au suma cifrelor un număr prim.
36. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se modifice ordinea valorilor din tablou, astfel încât toate valorile egale cu x să ocupe primele poziții din v , iar celelalte valori să se regăsească în continuarea acestora.
37. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se mute pe primele poziții elementele impare.
38. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se mute pe primele poziții elementele pătrate perfecte.
39. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se modifice ordinea valorilor din tablou, astfel încât toate valorile mai mari decât ultimul element din vector să ocupe primele poziții din v , iar celelalte valori să se regăsească în continuarea acestora.
40. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze înainte de elementul maxim din șir, o valoare x dată (vectorul conține un singur element maxim).
41. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze după elementul minim din șir, o valoare x dată (vectorul conține un singur element minim).
42. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze înainte de primul element par din șir, o valoare x dată.

43. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze după primul element impar din șir, o valoare x dată.
44. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze înainte de primul număr prim din șir, o valoare x dată.
45. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze după primul număr prim din șir, o valoare x dată.
46. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze înainte de fiecare termen par al șirului, numărul obținut prin împărțirea la 2 a valorii acestuia. Să se afișeze vectorul astfel obținut.
47. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze după fiecare termen impar al șirului, dublul acestuia. Să se afișeze vectorul astfel obținut.
48. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze după fiecare element nenul din tablou, câte un nou element, cu aceeași valoare absolută, dar cu semn opus. (Examen de Bacalaureat – Sesiunea Specială 2012 – enunț modificat)
49. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se insereze după fiecare element nenul din tablou, câte un nou element, cu aceeași valoare. (Examen de Bacalaureat - Iunie 2012 – enunț modificat)
50. Se citește un vector v cu n elemente numere întregi. Să se insereze între oricare 2 elemente alăturate media lor aritmetică.
51. Se citește un vector v cu n elemente numere întregi. Să se insereze între oricare două elemente numere prime, aflate pe poziții consecutive, media lor aritmetică.
52. Se citește un vector v cu n elemente numere întregi. Să se insereze între oricare 2 elemente alăturate, de semne diferite, modulul diferenței celor 2 numere.
53. Să se insereze într-un vector cu n elemente numere întregi, după fiecare element impar, dublul său.
54. Să se insereze într-un vector cu n elemente numere întregi, înaintea fiecărui element par, dublul său.
55. Se dau n numere întregi. Să se insereze între oricare două numere de aceeași paritate, partea întreagă a mediei lor aritmetice.
56. Scrieți un program care să realizeze inversarea unui vector în același vector, fără a utiliza un vector suplimentar.
57. Scrieți un program care să realizeze inversarea unui vector, folosind un vector suplimentar.
58. Se citește de la tastatură un vector v cu n elemente numere naturale, n număr par. Să se interschimbe primul element cu al doilea, al treilea cu al patrulea... penultimul cu ultimul și să se afișeze vectorul astfel obținut.
59. Să se înlocuiască toate elementele nule dintr-un vector cu elemente numere reale, cu media aritmetică a elementelor nenule din vector.

60. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se scrie un program care interschimbă două secvențe de lungimi n_1 și n_2 , care încep din pozițiile p_1 , respectiv p_2 . În cazul în care lungimile n_1 și n_2 sunt diferite, elementele care nu sunt cuprinse în cele două secvențe vor fi mutate, astfel încât să se poată face interschimbarea.
61. Determinați toate permutările circulare spre stânga cu o poziție ale unui vector dat. Permutările astfel obținute se vor afișa pe ecran, pe rânduri diferite.
62. Determinați toate permutările circulare spre dreapta cu o poziție ale unui vector dat. Permutările astfel obținute se vor afișa pe ecran, pe rânduri diferite.
63. Determinați toate permutările circulare spre stânga cu k poziții ale unui vector dat. Permutările astfel obținute se vor afișa pe ecran, pe rânduri diferite.
64. Determinați toate permutările circulare spre dreapta cu k poziții ale unui vector dat. Permutările astfel obținute se vor afișa pe ecran, pe rânduri diferite.

3.1.4. Perechi de numere. Triplete de numere.

1. Se citește de la tastatură un vector cu n elemente numere întregi. Să se afișeze toate perechile de elemente ale vectorului (nu neapărat consecutive), care au aceeași sumă a cifrelor.
2. Se citește de la tastatură un vector cu n elemente numere întregi. Să se precizeze de câte ori găsim elemente consecutive în vector care sunt egale între ele.
3. Fie v un vector cu n elemente numere întregi. Să se afișeze toate perechile de elemente consecutive din vector, cu proprietatea ca al doilea element al perechii este dublul primului.
4. Fie v un vector cu n elemente numere întregi. Să se afișeze toate perechile de elemente consecutive din vector, cu proprietatea ca al doilea element al perechii este mai mic decât primul element al perechii.
5. Să se afișeze toate perechile de elemente consecutive dintr-un vector cu n elemente numere întregi, care au proprietatea că sunt numere prime.
6. Se consideră un vector v cu n elemente numere întregi. Să se afișeze de câte ori găsim două elemente aflate pe poziții consecutive în vector, care să fie egale între ele.
7. Să se afișeze toate tripletele de elemente dintr-un vector cu n elemente numere întregi, care au proprietatea că sunt numere pitagoreice.
8. Să se afișeze toate tripletele de numere consecutive dintr-un vector v cu n elemente numere întregi cu proprietatea că elementul din mijloc este media aritmetică a celorlalte două.
9. Fie v un vector cu n elemente numere întregi. Să se afișeze toate tripletele de numere consecutive din v cu proprietatea că pot reprezenta laturile unui triunghi.

10. Fie v un vector cu n elemente numere întregi. Să se afișeze toate tripletele de numere consecutive din v cu proprietatea că pot reprezenta laturile unui triunghi isoscel.
11. Fie v un vector cu n elemente numere întregi. Să se afișeze pe ecran toate tripletele de numere naturale (x, y, z) cu proprietatea că $x < y < z$ și $x \cdot y + y \cdot z = n$. Fiecare triplet se afișează pe câte o linie a ecranului, iar numerele din fiecare triplet sunt separate prin câte o virgulă și încadrate între paranteze rotunde. (Exemplu $n = 8$ și $(1, 2, 3)$) (Examen de Bacalaureat - August 2014 - enunț modificat)
12. Fie un vector v cu n elemente numere întregi. Să se afișeze toate tripletele de numere consecutive din v în care al treilea număr este restul împărțirii primului la al doilea.
13. Fie v un vector cu n elemente numere întregi. Să se afișeze toate tripletele de numere consecutive din v , în care al treilea este media geometrică dintre primul și al doilea.

3.1.5. Verificarea unei proprietăți

1. Se consideră un șir cu n elemente numere naturale. Să se verifice dacă în șir există cel puțin un element impar.
2. Se consideră un șir cu n elemente numere întregi. Să se verifice dacă în șir există cel puțin un element pozitiv.
3. Se consideră un șir cu n elemente numere naturale. Să se verifice dacă toate elementele șirului au aceeași paritate.
4. Se consideră un vector cu n elemente numere naturale. Să se verifice dacă toate elementele vectorului sunt egale cu o valoare x dată.
5. Se consideră un șir cu n elemente numere naturale. Să se verifice dacă în șir există elemente numere prime.
6. Se citește un vector v cu n elemente numere naturale. Să se verifice dacă există cel puțin 3 elemente alăturate care să aibă aceeași paritate.
7. Se consideră un vector cu n elemente numere naturale. Să se verifice dacă are elementele ordonate crescător.
8. Se consideră un vector cu n elemente numere naturale. Să se verifice dacă are elementele ordonate descrescător.
9. Se consideră un vector cu n elemente numere naturale. Să se verifice dacă toate elementele vectorului sunt diferite două câte două.
10. Se consideră un șir cu n elemente numere întregi. Să se verifice dacă elementele impare din șir sunt ordonate crescător.
11. Să se verifice dacă un șir dat este *șir munte* (are elementele ordonare crescător până la o poziție k și descrescător de la poziția k până la sfârșit).
12. Să se verifice dacă un șir dat este *șir vale* (are elementele ordonare descrescător până la o poziție k și crescător de la poziția k până la sfârșit).

13. Se consideră un șir cu n elemente numere naturale. Să se verifice dacă toate elementele șirului au 3 cifre.
14. Un vector cu elemente 0 sau 1 se numește *alternativ* dacă oricum am lua două elemente vecine în vector, cel puțin unul dintre ele este 0. Se consideră un vector cu n elemente numere naturale. Să se verifice dacă vectorul este *alternativ*.
15. Se dau doi vectori cu câte n elemente numere naturale. Verificați dacă cei doi vectori sunt identici (conțin aceleași elemente).
16. Se consideră un vector cu n elemente numere naturale. Să se verifice dacă toate elementele vectorului aparțin unui interval $[a, b]$ cu a și b date de la tastatură.
17. Se consideră un șir cu n elemente numere naturale. Să se verifice dacă există în șir elemente care să aibă ambii vecini de aceeași paritate cu el. În cazul în care nu există nici un astfel de element se va afișa un mesaj corespunzător.
18. Se citește de la tastatură un vector cu n elemente numere întregi. Să se verifice dacă elementele vectorului sunt în progresie aritmetică, iar în caz afirmativ să se afișeze rația progresiei.
19. Se citește de la tastatură un vector cu n elemente numere întregi. Să se verifice dacă elementele vectorului sunt în progresie geometrică, iar în caz afirmativ să se afișeze rația progresiei.
20. Să se verifice dacă un șir cu n numere naturale este *oglinduit* (citit de la stânga la dreapta sau de la dreapta la stânga, este același șir).

3.1.6. Maxime și minime

1. Se citește un vector v cu n elemente numere întregi. Afișați valoarea maximă din vector.
2. Se citește un vector v cu n elemente numere întregi. Afișați valoarea minimă din vector și pozițiile pe care apare ea.
3. Să se determine indicele maximului și cel al minimului valorilor elementelor unui vector. În cazul în care există mai multe valori maxime (respectiv minime) se vor afișa toți indecșii pentru valorile maxime (respectiv minime).
4. Se citește de la tastatură un tablou unidimensional cu n elemente numere întregi. Să se afișeze elementul care apare de cele mai multe ori în tablou. Dacă există mai multe astfel de elemente, se vor afișa toate.
5. Se citește un vector v cu n elemente numere întregi. Să se afișeze elementul din vector care are număr maxim de divizori.
6. Fie v un vector cu n componente numere întregi. Să se determine cele mai mici 3 elemente pozitive din vector.

7. Să se determine cele mai mari k elemente dintr-un vector de numere întregi.
8. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se determine cel mai mic număr care se găsește pe poziții pare în vector.
9. Fie v un vector cu n elemente numere întregi. Să se determine cel mai mare număr rațional subunitar, în care numitorul și numărătorul sunt elemente ale vectorului.
10. Se citește un vector cu n elemente numere naturale. Să se determine câte elemente ale vectorului sunt egale cu jumătatea diferenței dintre cea mai mare și cea mai mică valoare din vector.
11. Se citesc de la tastatură 2 vectori u și v cu n și m elemente. Să se determine elementele din u care sunt mai mici decât toate elementele din v .
12. Se citește un vector cu n elemente, numere naturale. Să se determine suma valorilor elementelor cuprinse între cel mai mic și cel mai mare element pozitiv al vectorului.
13. Fie v un vector cu n elemente numere întregi. Să se precizeze dacă vectorul conține un număr *majoritar* (un număr este *majoritar* dacă numărul său de apariții este mai mare decât $n/2$).
14. Fie v un vector cu n elemente numere întregi. Să se afișeze elementul care apare de cele mai multe ori în vector. În cazul în care există mai multe astfel de elemente se vor afișa toate.
15. Fie v un vector cu n elemente numere întregi. Să se determine diferența maximă în modul dintre două elemente consecutive ale vectorului.
16. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se determine cel mai mare număr prim aflat pe poziții impare în vector. Dacă nu există un astfel de element, se va afișa un mesaj corespunzător.
17. Se consideră un vector cu n elemente numere întregi. Să se determine elementul maxim din subșirul format din termenii de rang par din vector.
18. Se consideră un vector cu n elemente numere întregi. Să se determine elementul minim din subșirul format din termenii de rang impar din vector.
19. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se determine elementul maxim din subșirul format din numerele pătrate perfecte de rang par ale șirului inițial și elementul minim din subșirul format din numerele pătrate perfecte de rang impar ale șirului inițial, precum și rangurile pe care apar aceste elemente. În cazul în care sunt mai multe numere care îndeplinesc condiția se vor afișa rangurile corespunzătoare pentru toate.
20. Se consideră coordonatele a n puncte din plan (x_i, y_i) , $(1 \leq i \leq n)$. Să se afișeze lungimea celui mai lung segment și coordonatele capetelor acestui segment. În cazul în care există mai multe segmente care să îndeplinească condiția, se vor afișa toate (Pentru calculul lungimii unui segment se va folosi formula distanței $AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$).

3.1.7. Sortări

1. Se citește un vector v cu n elemente cifre. Afișați cel mai mic număr natural care se poate forma cu toate cele n cifre din vectorul v .
2. Se citește un vector v cu n elemente numere naturale. Să se determine dacă vectorul v este o permutare a mulțimii $\{1, 2, \dots, n\}$.
3. Se consideră un vector cu n elemente numere naturale și un număr k . Ordonăți crescător primele k elemente ale vectorului și descrescător ultimele $n - k$ elemente.
4. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se transforme în mulțime, prin eliminarea elementelor care se repetă.
5. Se consideră un vector cu n elemente numere naturale. Afișați în ordine crescătoare valorile prime din vector.
6. Să se scrie un program care ordonează elementele unui vector descrescător după cifra unităților.
7. Să se scrie un program care ordonează elementele unui vector crescător după prima cifră a fiecărui număr (cifra unităților se consideră ultima cifră).
8. Se dau n numere naturale nenule. Ordonăți crescător cele n numere după numărul lor de divizori.
9. Se dau n numere naturale. Afișați aceste numere, ordonate descrescător după suma divizorilor.
10. Se dau înălțimile a n stâlpi numerotați de la 1 la n , exprimate prin numere naturale. Afișați numerele de ordine ale stâlpilor în ordinea crescătoare a înălțimii lor.
11. Se consideră un șir cu n elemente numere naturale nenule cu cel mult 4 cifre fiecare. Afișați, în ordine crescătoare, valorile pare din șir. Dacă șirul nu conține nici o valoare pară, se va afișa mesajul *Nu există*. (Variante Bacalaureat 2009)
12. Se consideră un vector cu n componente ($n \leq 30$). Pentru o valoare p dată ($1 < p < n$) ordonați crescător subsecvențele formate din primele p , respectiv ultimele $n - p$ elemente și apoi ordonați întregul vector prin interclasarea celor două subsecvențe.
13. Se consideră un vector cu n elemente numere naturale. Afișați în ordine crescătoare valorile din vector care sunt prime cu primul element al vectorului.
14. Se consideră un vector cu n elemente numere naturale distincte. Ordonăți descrescător elementele situate înaintea valorii minime din vector și crescător elementele situate după această valoare.
15. Să se ordoneze crescător elementele dintr-un șir dat, cuprinse între elementul de valoare maximă și cel de valoare minimă.
16. Se dă un vector cu n elemente, numere naturale. Afișați în ordine descrescătoare elementele. După fiecare element, se va insera indicele poziției pe care acesta se afla înainte ca vectorul să fie sortat.
17. Se dau cel mult 100000 de numere naturale, cu cel mult 2 cifre fiecare. Afișați în ordine strict crescătoare valorile impare care se regăsesc printre valorile date și în ordine strict descrescătoare valorile pare care se regăsesc printre valorile date. (Variante Bacalaureat 2009)
18. Să se ordoneze crescător elementele pătrate perfecte ale unui șir dat, fără a afecta elementele care nu sunt pătrate perfecte.
19. Se citesc de la tastatură n elemente ale unui vector alcătuit exclusiv din litere mici și cifre. Rearanjați vectorul astfel încât cifrele să fie plasate pe primele poziții. Celelalte caractere își vor păstra ordinea inițială, de la stânga la dreapta.
20. Fie dat un număr natural n , să se determine cel mai mare număr care se poate forma folosind cifrele lui n . Se va folosi un vector care va memora cifrele lui n .
21. Se citește de la tastatură un număr natural n , apoi n numere naturale. Să se afișeze cel mai mic număr care poate fi scris folosind cifra maximă a fiecărui număr citit.
22. Să se scrie un program care ordonează crescător elementele situate pe poziții pare într-un vector și descrescător elementele situate pe poziții impare.
23. Se citește de la tastatură un număr natural n , apoi n numere naturale. Să se afișeze cel mai mare număr care poate fi scris folosind cifra (cifrele) din mijloc a numerelor citite.
24. Să se scrie un program care ordonează elementele unui vector crescător după numărul cifrelor elementelor. În cazul în care există mai multe numere cu același număr de cifre nu se va lua în considerare ordinea.
25. Se dau n numere naturale. Afișați în ordine crescătoare acele numere date care sunt puteri ale lui 2.
26. Se dau n numere naturale și un număr natural k . Afișați în ordine crescătoare acele numere date care sunt puteri ale lui k . (Variante Bacalaureat 2009)
27. Se dă un vector cu n elemente, numere naturale. Afișați în ordine descrescătoare valorile din vector care nu sunt prime cu primul element al vectorului.
28. Se citește un șir cu n numere întregi. Să se rearanjeze elementele șirului astfel ca numerele negative să fie ordonate descrescător. Apoi să urmeze elementele nule, urmate de numerele pozitive ordonate crescător.
29. Se dau două mulțimi prin intermediul a doi vectori u și v cu n componente numere întregi. Să se verifice dacă aceste mulțimi sunt proporționale (elementele lor formează proporții). În caz afirmativ se va afișa șirul de rapoarte egale.

30. Se consideră un vector cu n elemente numere întregi. Știind că el conține două subsecvențe de numere ordonate crescător, să se ordoneze întregul vector prin interclasarea celor două subsecvențe.
31. Se consideră două șiruri $a_1, a_2, a_3, \dots, a_n$ și $b_1, b_2, b_3, \dots, b_n$ ordonate crescător. Să se verifice dacă unul dintre șiruri este permutare circulară a celuilalt.
32. Se consideră un vector conținând doar elemente de 0 și 1. Să se ordoneze crescător acest vector, parcurgându-l o singură dată și fără a folosi vectori auxiliari.
33. Se citește de la tastatură n numere naturale ($n \leq 30$). Se cere să se ordoneze șirul crescător după numărul de apariții a cifrei 1 în reprezentarea în baza 2 a numerelor. În cazul în care două numere au același număr de cifre egale cu 1 în reprezentarea lor binară atunci ele vor apărea ordonate descrescător.
34. Se consideră n intervale de numere întregi $[a_i, b_i], 1 \leq i \leq n$. Să se determine numărul maxim de intervale care se *suprapun* (au cel puțin o valoare comună).
35. Se dau doi vectori u și v cu n și m elemente, reprezentând elementele a două mulțimi. Să se calculeze intersecția, reuniunea și diferența celor două mulțimi. Cei doi vectori se consideră sortați crescător.
36. Se consideră șirurile $s_1, s_2, s_3, \dots, s_n$ de lungimi $l_1, l_2, l_3, \dots, l_n$. Elementele fiecărui șir sunt ordonate crescător. Să se obțină, folosind interclasarea, un șir s care să conțină toate elementele celor n șiruri în ordine crescătoare.

3.1.8. Subșiruri

1. Se consideră un vector format din n elemente, numere naturale. Calculați suma elementelor din secvența determinată de primul și ultimul element par.
2. Se consideră un șir format din n elemente, numere naturale. Calculați suma elementelor din secvența determinată de primul și ultimul element prim.
3. Se consideră un vector v cu n elemente numere naturale de cel puțin 2 cifre fiecare. Să se determine cea mai lungă subsecvență de elemente consecutive prime, care au inversele tot numere prime.
4. Să se determine cea mai lungă secvență de elemente pare dintr-un vector.
5. Să se determine cea mai lungă secvență de elemente ordonate strict crescător dintr-un vector.
6. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se afișeze subșirul de lungime maximă care începe și se termină cu același număr.
7. Se consideră un șir cu n elemente, numere naturale. Determinați cea mai lungă secvență de elemente din șir cu proprietatea că oricare două valori consecutive în secvență au parități diferite.

8. Se consideră un șir cu elemente numere naturale nenule. Să se afle câte secvențe de numere consecutive din șir au suma mai mică decât un număr x dat.
9. Se consideră un vector cu n elemente numere naturale. Determinați cea mai lungă secvență de elemente nule din vector. Dacă în vector există mai multe secvențe de elemente nule de lungime maximă se va determina cea mai din stânga.
10. Se consideră un vector cu n elemente numere naturale. Determinați cea mai lungă secvență de elemente nenule din vector. Dacă în vector există mai multe secvențe de elemente nenule de lungime maximă se va determina cea mai din dreapta.
11. Se consideră un vector cu n elemente, numere naturale. Determinați cea mai lungă secvență de elemente egale din vector. Dacă în vector există mai multe secvențe de elemente egale de lungime maximă se va determina cea mai din dreapta.
12. Se consideră un vector v cu n elemente numere întregi și un vector u cu m elemente, numere întregi. Să se verifice dacă vectorul u este subșir al vectorului v .
13. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se determine secvența maximă din vector care are elementele consecutive prime între ele (x și y sunt prime între ele dacă $\text{cmmdc}(x, y) = 1$). Dacă problema are mai multe soluții se va afișa o singură soluție.
14. Se consideră un vector cu n elemente, numere naturale și un număr k . Să se stabilească dacă în vector există o secvență de lungime k cu elemente identice.
15. Se consideră un vector cu n elemente, numere naturale. Determinați câte secvențe ale vectorului au toate elementele egale (O secvență va avea minim 3 elemente).
16. Fie v un vector cu n elemente numere întregi. Să se determine o secvență de elemente consecutive din vector pentru care suma elementelor este divizibilă cu un număr k dat. În cazul în care nu există o astfel de secvență se va afișa un mesaj corespunzător.
17. Să se scrie un program care, pentru două tablouri unidimensionale date, A și B , verifică dacă tabloul A se poate reduce la tabloul B . (Variante Bacalaureat 2009).
18. Se dă un vector format din n elemente, numere naturale nenule și un număr natural S . Determinați, dacă există o secvență de elemente din șir cu suma elementelor egală cu S .
19. Se consideră un șir cu n elemente, numere întregi. Determinați secvența de elemente cu suma maximă.
20. Se consideră un vector cu n elemente, numere naturale. Determinați secvența palindromică de lungime maximă.

21. Prin *pantă* înțelegem o secvență de elemente consecutive dintr-un vector v pentru care $v_i = v_{i-1} + 1$. Să se determine panta de lungime maximă.
22. Se consideră un vector cu n elemente numere întregi. Se cere să se determine o succesiune cu număr maxim de elemente din șir, cu proprietatea că fiecare componentă a succesiunii are același număr de cifre distincte în scrierea sa.
23. Numim *secvență încadrată* a unui șir de numere naturale un subșir al acestuia, format din termeni aflați pe poziții consecutive în șirul dat, subșir care începe și se termină cu aceeași valoare. Lungimea secvenței este egală cu numărul de termeni ai acesteia.
Să se determine secvențele încadrate dintr-un șir, care au lungimea maximă. (Examen de Bacalaureat – Model 2018)

3.1.9. Vectori caracteristici

1. Se consideră un șir cu n numere naturale de două cifre. Să se determine numerele naturale nenule de două cifre care nu apar în șirul dat.
2. Se dau mai multe numere naturale formate din exact o cifră. Determinați cifrele cu număr maxim de apariții.
3. Să se scrie un program care citește cel mult 1000000 de numere naturale din intervalul închis $[0,9]$ și determină cel mai mare număr prim citit și numărul său de apariții. (Variante Bacalaureat 2009)
4. Se dau mai multe numere naturale cu cel mult 9 cifre fiecare. Să se determine cele mai mari două numere de 3 cifre care nu se găsesc printre numerele date. (Variante Bacalaureat 2009)
5. Se dau mai multe numere naturale, fiecare cu cel mult 9 cifre. Să se afișeze, în ordine descrescătoare, toate cifrele care apar în numerele date. (Variante Bacalaureat 2009)
6. Se citește un număr natural n . Să se afișeze cel mai mic număr care se poate forma cu cifrele numărului n .
7. Un număr natural x , format din exact două cifre, este numit *sub-număr* al unui număr natural y dacă cifrele lui x apar, în aceeași ordine, pe ranguri consecutive, în numărul y .
Să se determine, pentru mai multe numere date, *sub-numerele* care apar de cele mai multe ori în scrierea acestora. (Examen de Bacalaureat - Iunie 2014)
8. Se dau n numere naturale cu cel mult două cifre fiecare. Să se determine acele numere care apar o singură dată. (Variante Bacalaureat 2009)
9. Se consideră un vector cu n elemente, numere naturale nenule. Afișați termenii în ordine crescătoare. Termenii care apar de mai multe ori se vor afișa o singură dată.

10. Se dau n numere naturale cu cel mult două cifre fiecare. Afișați valorile distincte în ordinea crescătoare a numărului de apariții.
11. Fiind date două numere a și b să se precizeze câte cifre au în comun.
12. Afișați cel mai mare număr care se poate forma cu cifrele distincte ale unui număr x dat.
13. Un șir de numere este o *progresie aritmetică de rație r* dacă oricare termen al său, cu excepția primului, se obține din cel care îl precede, prin adunarea la acesta a numărului r . Exemplu: șirul 12, 14, 16, 18, 20 este o progresie de rație 2. Se citește de la tastatură un vector cu elemente. Se cere să se verifice dacă există un număr natural r , astfel încât toate numerele distincte din șir să poată fi rearanjate, pentru a forma o progresie aritmetică de rație r . Se afișează pe ecran numărul r , sau mesajul NU, dacă nu există un astfel de număr. (Examen Bacalaureat – August 2017 enunț modificat)

3.2 Tablouri bidimensionale (Matrice)

3.2.1. Parcurgeri

1. Se consideră o matrice A cu n linii și m coloane și elemente numere naturale. Să se calculeze suma elementelor matricei.
2. Se consideră o matrice A cu n linii și m coloane și elemente numere naturale. Să se calculeze suma valorilor pare din matrice.
3. Se consideră o matrice A cu n linii și m coloane și elemente numere naturale. Să se calculeze suma elementelor matricei de pe linii pare și coloane impare.
4. Se consideră o matrice A cu n linii și m coloane și elemente numere întregi. Să se calculeze suma elementelor pozitive și impare ale matricei.
5. Se citește de la tastatură o matrice A cu m linii și n coloane cu elemente numere întregi. Să se afișeze suma elementelor negative de pe coloane pare din matrice.
6. Fiind dată o matrice A cu m linii și n coloane cu elemente numere întregi, să se afișeze numărul elementelor pozitive, numărul elementelor negative și numărul elementelor nule din matrice.
7. Fie A o matrice cu m linii și n coloane. Să se scrie un program care înmulțește cu o constantă k fiecare element al tabloului.
8. Fiind dată o matrice A cu m linii și n coloane, să se scrie un program care determină media aritmetică a elementelor negative ale matricei.
9. Fie A o matrice cu m linii și n coloane. Să se scrie un program care afișează suma elementelor pe linii.
10. Se citește de la tastatură o matrice A cu $m \times n$ elemente numere întregi. Să se calculeze suma elementelor de pe liniile pare și să se afișeze.

11. Fie A o matrice cu m linii și n coloane cu elemente numere întregi. Să se scrie un program care afișează liniile din matrice care conțin k elemente nule (k este dat de la tastatură).
12. Se citește de la tastatură o matrice a cu n linii și n coloane și elemente numere întregi. Să se determine perechile de indici (i, j) pentru care suma vecinilor lui a_{ij} este pozitivă (se vor lua în considerare decât vecinii de pe linie).
13. Se citește de la tastatură o matrice a cu n linii și n coloane și elemente numere întregi. Să se determine perechile de indici (i, j) pentru care suma vecinilor lui a_{ij} este pozitivă (se vor lua în considerare toți vecinii, de pe linii și coloane).
14. Se citește de la tastatură o matrice a cu n linii și n coloane și elemente numere întregi. Să se determine perechile de indici (i, j) pentru care suma vecinilor lui a_{ij} este maximă.
15. Pentru o matrice A cu m linii și n coloane să se afișeze coordonatele tuturor *vârfurilor*. Un *vârf* este un element al matricei care are toți vecinii mai mici decât el.
16. Scrieți un program care citește de la tastatură un tablou bidimensional cu n linii și n coloane și afișează pe ecran numărul valorilor din tablou care sunt strict mai mici decât toți vecinii lor.
17. Dându-se o matrice cu $n \times n$ elemente, se cere să se formeze un vector, prin parcurgerea matricei sub forma de L . Exemplu: Dacă $n = 3$ matricea va fi parcursă astfel: $a_{11}, a_{21}, a_{31}, a_{32}, a_{33}, a_{12}, a_{22}, a_{23}, a_{13}$.
18. Se citește de la tastatură o matrice a cu n linii și n coloane cu elemente numere întregi. Să se afișeze elementele de pe pătratele concentrice ale matricei. Elementele de pe un pătrat se vor afișa pe un rând.
19. Fie A o matrice cu m linii și n coloane. Să se verifice dacă ea conține cel puțin o linie simetrică (elementele egal depărtate de capetele respective sunt egale).
20. Fiind dată o matrice A cu m linii și n coloane, se cere să se calculeze suma componentelor numere prime ce se află pe pătratele concentrice ale tabloului, dinspre exterior spre interior.
21. Scrieți un program care citește de la tastatură un număr natural n ($2 < n < 100$) și apoi n linii cu câte n numere naturale de cel mult 5 cifre ce formează un tablou bidimensional A . Să se afișeze pe ecran, separate printr-un spațiu, elementele tabloului A care au proprietatea că divid suma vecinilor existenți (pe linie, coloană și diagonale). Elementele vor fi identificate prin parcurgerea tabloului A pe linii, de sus în jos, pe fiecare linie parcurgerea făcându-se de la stânga la dreapta. (Variante Bacalaureat 2009)

22. Scrieți un program care citește de la tastatură două numere naturale m și n ($2 \leq m \leq 20, 2 \leq n \leq 20$) și construiește în memorie un tablou bidimensional A , cu m linii și n coloane, astfel încât parcurgându-l linie cu linie, de jos în sus, și fiecare linie de la dreapta la stânga, să se obțină șirul primelor $n \times m$ numere naturale, impare, care *NU* sunt divizibile cu 3, ordonat strict crescător. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele de pe aceeași linie fiind separate prin câte un spațiu. (Examen de Bacalaureat – Model 2015)
23. Să se verifice dacă o matrice pătratică A de ordin n este *pătrat magic* sau nu. O matrice pătratică este *pătrat magic* dacă sumele de pe fiecare linie, coloană și de pe cele două diagonale sunt egale.
24. O metodă de *generare a unui pătrat magic* constă în completarea pe rând cu valorile $1, 2, \dots, n^2$ după următoarea regulă:
 - Se plasează 1 în centrul ultimei coloane;
 - Se merge în line oblică spre dreapta-jos, cu numărul următor; dacă se iese din pătrat prin partea dreaptă, se merge în partea opusă, în stânga liniei unde trebuia așezat numărul, iar dacă se iese prin partea de jos a pătratului, se merge în partea de sus a coloanei unde trebuia așezat numărul;
 - După ce se completează n numere, se merge mai jos cu o poziție spre stânga, pe aceeași linie, pentru a se repeta apoi pasul 2 și a genera apoi următorul grup de numere.
 Să se scrie un program pentru generarea unui pătrat magic.
25. Se consideră o matrice A cu m linii și n coloane cu elemente de 0 și 1. Să se scrie un program care să afișeze toate perechile de *linii complementare*. Două linii se numesc *complementare* dacă suma componentelor de pe aceeași coloană este 1.
26. Se consideră o matrice A cu m linii și n coloane cu elemente numere întregi. Se cere să se calculeze suma elementelor de pe *marginea tabloului* (formată din prima linie, ultima linie, prima coloană și ultima coloană).
27. Se consideră o matrice cu n linii și m coloane și elemente numere naturale. Să se determine mulțimea formată din elementele distincte ale chenarului (marginea) matricei.
28. Scrieți un program care citește de la tastatură numere naturale din intervalul $[3, 10^2]$, în această ordine: n și m , apoi elementele unui tablou bidimensional cu n linii și m coloane, iar la final un număr x . Programul afișează pe ecran mesajul *DA*, dacă există cel puțin un element egal cu x aflat pe *conturul tabloului* (format din prima linie, ultima linie, prima coloană și ultima coloană), sau mesajul *NU* în caz contrar. (Examen de Bacalaureat - August 2016)
29. Scrieți un program care citește de la tastatură numere naturale din intervalul $[3, 10^2]$, în această ordine: n și m , apoi elementele unui tablou bidimensional cu n linii și m coloane. Să se afișeze toate elementele pătrate perfecte de pe conturul tabloului.

30. Scrieți un program care citește de la tastatură două numere naturale din intervalul $[3, 50]$, n și m , și elementele unui tablou bidimensional cu n linii și m coloane, numere naturale din intervalul $[0, 10^4]$. Programul modifică în memorie tabloul dat, atribuind valoarea elementului aflat pe ultima linie și pe ultima coloană a tabloului, fiecărui element aflat pe conturul acestuia (pe prima linie, ultima linie, prima coloană, ultima coloană), apoi afișează pe ecran tabloul modificat, câte o linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spațiu. (Examen de Bacalaureat - Iunie 2017)
31. Scrieți un program care citește de la tastatură numere naturale din intervalul $[3, 10^2]$, în această ordine: n și m , apoi elementele unui tablou bidimensional cu n linii și m coloane, iar la final un număr x . Programul afișează pe ecran mesajul *DA*, dacă există cel puțin un element egal cu x aflat pe conturul tabloului (format din prima linie, ultima linie, prima coloană și ultima coloană), sau mesajul *NU* în caz contrar. (Examen de Bacalaureat - August 2016)
32. Să se scrie un program care calculează transpusa unei matrice A cu $n \times m$ elemente.
33. Să se scrie un program care calculează produsul a două matrice pătratice A și B cu $n \times n$ elemente numere întregi.
34. Să se parcurgă în sensul acelor de ceasornic chenarul exterior al unei matrice pătratice. (Variante Bacalaureat 2009)
35. Să se parcurgă o matrice pătratică în *spirală*, în sensul acelor de ceasornic. Să se afișeze elementele obținute în urma acestei parcurgeri.
36. Se consideră o matrice cu n linii și n coloane și elemente numere întregi. Valoarea lui n este un număr impar. Putem împărți o astfel de matrice în patru cadrane, prin intermediul liniei și coloanei din mijloc. Să se calculeze și să se afișeze sumele din fiecare cadran. Sumele cerute nu vor conține și valorile de pe linia și coloana din mijloc.
37. Se citește de la tastatură o matrice A cu n linii și n coloane cu elemente numere întregi. Să se calculeze A^2 .
38. Se citește de la tastatură o matrice A cu n linii și n coloane cu elemente numere întregi. Să se calculeze A^m (m este un număr natural dat, $m > 2$).
39. Fie A o matrice pătratică cu $n \times n$ elemente numere întregi. Să se determine matricea $B = A \cdot A^t$, unde A^t reprezintă transpusa matricei A .

3.2.2. Construcții matrice și vectori

1. Se citește de la tastatură o matrice A cu n linii și n coloane cu elemente numere întregi. Să se copieze într-un vector elementele pozitive ale matricei, apoi să se afișeze vectorul astfel obținut.
2. Se citește de la tastatură o matrice A cu n linii și n coloane cu elemente numere întregi. Să se copieze într-un vector elementele numere prime ale matricei, apoi să se afișeze vectorul astfel obținut.

3. Se citește de la tastatură o matrice A cu n linii și n coloane cu elemente numere întregi. Să se copieze într-un vector sumele elementelor de pe fiecare linie.
4. Se citește de la tastatură o matrice A cu n linii și n coloane cu elemente numere întregi. Să se copieze într-un vector sumele elementelor de pe fiecare coloană.
5. Se citește de la tastatură o matrice A cu n linii și n coloane cu elemente numere întregi. Să se copieze într-un vector sumele elementelor de pe liniile pare.
6. Se citește de la tastatură o matrice A cu n linii și n coloane cu elemente numere întregi. Să se copieze într-un vector sumele elementelor de pe coloanele impare.
7. Să se construiască o matrice A cu n linii și n coloane care să conțină primele n^2 numere pare. Completarea se va face pe linii.
8. Să se construiască o matrice A cu n linii și n coloane care să conțină primele n^2 numere prime. Completarea se va face pe linii.
9. Să se construiască o matrice A cu n linii și n coloane care să conțină primele n^2 numere pătrate perfecte. Completarea se va face pe linii.
10. Să se construiască o matrice A cu n linii și n coloane care să conțină primii n^2 termeni din *șirul lui Fibonacci*. Completarea se va face pe linii.
11. Scrieți un program care citește de la tastatură un număr natural n și construiește în memorie o matrice cu n linii și n coloane ale cărei elemente sunt numere naturale, fiecare reprezentând ultima cifră a câte unui termen al *șirului lui Fibonacci*, începând de la termenul de indice 1 și până la termenul de indice n^2 . (Variante Bacalaureat 2009)
12. Să se completeze în spirală o matrice A cu n linii și n coloane cu primii n^2 termeni ai *șirului lui Fibonacci*.
13. Scrieți un program care citește de la tastatură două numere naturale m și n , din intervalul $[2, 50]$ și construiește în memorie un tablou bidimensional cu m linii și n coloane, astfel încât prima coloană conține primele m numere naturale impare, în ordine strict crescătoare, și parcurgând orice linie a tabloului, de la stânga la dreapta, se obțin numere naturale consecutive, în ordine strict crescătoare. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele de pe aceeași linie fiind separate prin câte un spațiu. (Examen de Bacalaureat - Simulare 2015)
14. Scrieți un program care citește de la tastatură un număr natural nenul n și construiește un tablou bidimensional de dimensiune n în care fiecare element este egal cu ultima cifră a produsului indicilor. (Variante Bacalaureat 2009)
15. Scrieți un program care citește de la tastatură un număr natural nenul n cu cel mult 9 cifre și construiește un tablou bidimensional pătratic cu dimensiunea egală cu numărul de cifre ale lui n , completată cu cifrele lui n (Elementele de pe prima coloană vor fi completate cu cifra unităților lui n , cele de pe a doua coloană cu cifra zecilor lui n ș.a.m.d.) (Variante Bacalaureat 2009)

16. Scrieți un program care citește de la tastatură un număr natural n și construiește în memorie o matrice cu n linii și n coloane, numerotate de la 1 la n , în care elementele de pe liniile cu indice impar sunt egale cu indicele de linie, iar elementele situate pe linii cu indice par sunt egale cu indicele de coloană. (Variante Bacalaureat 2009)
17. Scrieți un program care citește de la tastatură două numere naturale nenule n și m , apoi construiește în memorie o matrice cu n linii și m coloane astfel încât, parcurgând tabloul linie cu linie de sus în jos și fiecare linie de la stânga la dreapta, să se obțină șirul primelor $n \times m$ pătrate perfecte impare, ordonat strict crescător. (Variante Bacalaureat 2009)
18. Scrieți un program care citește de la tastatură un număr natural n și construiește în memorie o matrice cu n linii și n coloane, numerotate de la 1 la n , în care fiecare element aflat pe chenarul exterior al matricei este egal cu suma dintre indicele liniei și indicele coloanei pe care se află, iar fiecare dintre celelalte elemente este egal cu suma celor trei vecini situați în matrice pe linia anterioară. (Variante Bacalaureat 2009)
19. Scrieți un program care citește de la tastatură un număr natural n și construiește în memorie o matrice cu n linii și n coloane, numerotate de la 1 la n , în care se vor memora în ordinea crescătoare a valorii, pe linii și coloane, primele $n \times n$ numere naturale nenule pare, care nu sunt divizibile cu 3. (Variante Bacalaureat 2009)
20. Scrieți un program care citește de la tastatură două numere naturale nenule n și m și care construiește în memorie și apoi afișează o matrice A cu n linii și m coloane cu proprietatea că fiecare element a_{ij} memorează cea mai mică dintre valorile indicilor i și j . (Variante Bacalaureat 2009)
21. Se citesc de la tastatură 2 numere naturale nenule m, n ($2 < m, n < 10$). Să se scrie un program care construiește în memorie o matrice A cu m linii (numerotate de la 1 la m) și n coloane (numerotate de la 1 la n), cu proprietatea că $A[i][j]$ este cel mai mic număr care se poate obține prin concatenarea lui i cu j . Matricea se va afișa pe ecran, câte o linie a matricei pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin spații. (Variante Bacalaureat 2009)
22. Se citesc de la tastatură 2 numere naturale nenule m și n ($2 < m, n < 20$). Să se scrie un program care construiește în memorie o matrice A cu m linii (numerotate de la 1 la m) și n coloane (numerotate de la 1 la n) cu proprietatea că elementul $A[i][j]$ este egal cu cel mai mare divizor comun dintre numerele i și j . Matricea se va afișa pe ecran, câte o linie a matricei pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin spații. (Variante Bacalaureat 2009)
23. Scrieți un program care citește de la tastatură două numere naturale nenule n și m și care construiește în memorie, apoi afișează o matrice A cu n linii și m coloane astfel încât prin parcurgerea acesteia linie cu linie de sus în jos și fiecare linie de la stânga la dreapta, se obțin în ordine descrescătoare toate numerele naturale de la 1 la $m \times n$. (Variante Bacalaureat 2009)

24. Scrieți un program care citește de la tastatură un număr natural n și construiește în memorie o matrice cu n linii și n coloane, numerotate de la 1 la n , în care se vor memora în ordinea crescătoare a valorii, pe linii și coloane, primele $n \times n$ numere naturale nenule, pare, care nu sunt divizibile cu 3. (Variante Bacalaureat 2009)
25. Scrieți un program care citește de la tastatură două numere naturale n și k și construiește în memorie o matrice cu n linii și n coloane formată numai din valori 1, 2, 3 și 4 astfel încât: elementele aflate la intersecția primelor k linii cu primele k coloane sunt egale cu 1, elementele aflate la intersecția primelor k linii cu ultimele $n - k$ coloane sunt egale cu 2, elementele aflate la intersecția ultimelor $n - k$ linii cu primele k coloane sunt egale cu 3, elementele aflate la intersecția ultimelor $n - k$ linii cu ultimele $n - k$ coloane sunt egale cu 4. (Variante Bacalaureat 2009)
26. Scrieți un program care citește de la tastatură trei numere naturale n, a, b și construiește în memorie o matrice cu n linii și n coloane, formată numai din valori 0, 1, 2, 3 și 4, astfel încât: elementele aflate pe linia a sau coloana b sunt egale cu 0, cele aflate deasupra liniei a și la stânga coloanei b sunt egale cu 1, cele aflate deasupra liniei a și la dreapta coloanei b sunt egale cu 2, cele aflate sub linia a și la stânga coloanei b sunt egale cu 3, iar elementele aflate sub linia a și la dreapta coloanei b sunt egale cu 4. (Variante Bacalaureat 2009)
27. Scrieți un program care citește de la tastatură două numere naturale n și m și construiește în memorie o matrice cu n linii și m coloane care conține numerele naturale de la 1 la $m \times n$, dispuse astfel:
- prima linie conține numerele de la 1 la m , ordonate crescător;
 - a doua linie conține numerele de la $m + 1$ la $2 \times m$, ordonate descrescător;
 - a treia linie conține numerele de la $2 \times m + 1$ la $3 \times m$, ordonate crescător;
 - a patra linie conține numerele de la $3 \times m + 1$ la $4 \times m$, ordonate descrescător;
 - etc. (Variante Bacalaureat 2009)
28. Scrieți un program care citește de la tastatură un număr natural n și construiește în memorie o matrice cu n linii și n coloane în care:
- toate elementele de pe prima coloană au valoarea 1;
 - ultima linie conține, în ordine strict crescătoare, numerele naturale din intervalul $[1, n]$;
 - oricare alt element este obținut prin însumarea celor două elemente vecine cu el, aflate pe linia imediat următoare și pe aceeași coloană cu el, respectiv pe aceeași linie cu el și pe coloana anterioară. (Examen de Bacalaureat – August 2012)
29. Scrieți un program care citește de la tastatură un număr natural n și construiește în memorie o matrice pătratică cu $n \times n$ elemente în care fiecare linie a matricei să conțină o permutare a mulțimii $\{1, 2, \dots, n\}$, astfel încât pe linii diferite ale matricei să se afle permutări diferite. Prima linie va conține permutarea identică $\{1, 2, 3, \dots, n\}$.

30. Scrieți un program care citește de la tastatură două numere naturale m și n ($2 \leq m \leq 20$, $2 \leq n \leq 20$) și construiește în memorie un tablou bidimensional A , cu m linii și n coloane, astfel încât parcurgându-l linie cu linie, de jos în sus, și fiecare linie de la dreapta la stânga, să se obțină șirul primelor $n \times m$ numere naturale impare, care NU sunt divizibile cu 3, ordonat strict crescător. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele de pe aceeași linie fiind separate prin câte un spațiu. (Examen de Bacalaureat - Model 2015)

31. Scrieți un program care citește de la tastatură două numere naturale, m și n ($2 \leq m \leq 20$, $2 \leq n \leq 20$), și construiește în memorie un tablou bidimensional A , cu m linii și n coloane, astfel încât parcurgându-l linie cu linie, de sus în jos, și fiecare linie de la stânga la dreapta, să se obțină șirul primelor $n \times m$ numere naturale, pare, care NU sunt divizibile cu 5, ordonat strict crescător. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele de pe aceeași linie fiind separate prin câte un spațiu. (Examen de Bacalaureat - August 2014)

32. Scrieți un program care citește de la tastatură un număr natural n ($n \in [2, 10^2]$) și construiește în memorie un tablou bidimensional cu n linii și n coloane, astfel:

- prima coloană conține, în ordine strict crescătoare, numerele naturale din intervalul $[1, n]$;
- toate elementele ultimei linii au valoarea n ;
- oricare alt element este obținut prin însumarea celor două elemente vecine cu el, aflate pe coloana anterioară, unul pe aceeași linie cu el, iar celălalt pe linia următoare;

Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spațiu. (Examen de Bacalaureat - Iunie 2016)

33. Scrieți un program care citește de la tastatură două numere naturale din intervalul $[3, 50]$, n și m și elementele unui tablou bidimensional cu n linii și m coloane, numere naturale din intervalul $[0, 10^4]$. Programul modifică în memorie tabloul dat, atribuind valoarea elementului aflat pe ultima linie și pe ultima coloană a tabloului fiecărui element aflat pe conturul acestuia (pe prima linie, ultima linie, prima coloană, ultima coloană), apoi afișează pe ecran tabloul modificat, câte o linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spațiu. (Examen de Bacalaureat - Iunie 2017)

34. Scrieți un program care construiește în memorie o matrice pătratică de dimensiune n ($1 < n < 30$) care să conțină pe prima linie, în ordine crescătoare, numerele de la 1 la n , pe a doua linie, în ordine descrescătoare, numerele de la $2n$ la $n + 1$, pe a treia linie, în ordine crescătoare, numerele de la $2n + 1$ la $3n$ etc. Valoarea lui n se citește de la tastatură, iar matricea se va afișa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu spații între elementele fiecărei linii.

35. Scrieți un program care să genereze o matrice pătratică cu n linii și n coloane și elemente numere naturale nenule cu proprietățile:

- Suma elementelor matricei este egală cu o valoare dată S ;
- Nu există elemente identice pe o linie sau pe o coloană;
- Diferența dintre cel mai mare și cel mai mic element al matricei este minimă.

Valorile lui n și S se citesc de la tastatură. Matricea va fi afișată pe ecran, elementele fiecărei linii a matricei se scriu pe câte un rând cu spații între ele.

3.2.3. Operații cu matrice

- Să se elimine dintr-o matrice A cu m linii și n coloane linia l și coloana c , apoi să se afișeze matricea rămasă (nu se va folosi o matrice auxiliară).
- Fiind dată o matrice A cu m linii și n coloane, să se elimine liniile și coloanele la a căror intersecție se găsește valoarea x dată de la tastatură.
- Se citește de la tastatură o matrice cu n linii și n coloane cu elemente numere întregi. Să se șteargă din matrice liniile și coloanele pe care se află cele mai mici două elemente de pe diagonala secundară. Să se afișeze matricea astfel rezultată.
- Se citește de la tastatură o matrice A cu $m \times n$ elemente numere întregi. Să se interschimbe între ele două linii date L_1 și L_2 . Nu se va folosi altă matrice.
- Să se insereze o linie l în poziția p și o coloană c în poziția k , elementele acestora fiind citite de la tastatură.
- Se consideră o matrice A cu m linii și n coloane și un număr k . Să se alcătuiască un program care transferă primele k coloane la sfârșitul tabloului.
- Să se bordeze o matrice A cu m linii și n coloane cu linia $m + 1$ și coloana $n + 1$, unde $a_{m+1,j}$ să fie suma elementelor de pe coloana j , iar $a_{i,n+1}$ să fie suma elementelor de pe linia i . Ultimul element va fi dat de suma elementelor de pe diagonala principală.
- Se dă o matrice cu n linii și m coloane cu elemente numere naturale. Să se permute coloanele matricei circular spre stânga cu o poziție.
- Se dă o matrice cu n linii și m coloane cu elemente numere naturale. Să se permute coloanele matricei circular spre dreapta cu o poziție.
- Se dă o matrice cu n linii și m coloane cu elemente numere naturale. Să se permute liniile matricei circular în jos cu o poziție.
- Se dă o matrice cu n linii și m coloane cu elemente numere naturale. Să se permute liniile matricei circular în sus cu o poziție.
- Se dă o matrice A cu m linii și n coloane cu elemente numere întregi. Să se permute circular cu k poziții în jos liniile lui A .

13. Se dă o matrice A cu m linii și n coloane cu elemente numere întregi. Să se permută circular cu k poziții spre dreapta coloanele lui A .
14. Pentru o matrice dată A cu $m \times n$ elemente numere întregi, să se elimine liniile și coloanele la intersecția cărora se află valoarea 0. Nu se va folosi altă matrice. Programul va afișa matricea rezultată după eliminări.
15. Scrieți un program care citește de la tastatură două numere naturale, m și n ($3 \leq m \leq 50, 3 \leq n \leq 50$) și elementele unui tablou bidimensional cu m linii și n coloane, numere naturale cu cel mult patru cifre, apoi modifică tabloul în memorie, eliminând penultima linie și penultima coloană a acestuia. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele de pe aceeași linie fiind separate prin câte un spațiu. (Examen de Bacalaureat - Iunie 2014)
16. Se citește de la tastatură o matrice A cu $m \times n$ elemente numere întregi și patru valori întregi x, y, l, c . Să se insereze în matricea A
 - Pe linia l , o linie alcătuită din valoarea x ;
 - Pe coloana c , o coloană alcătuită din valoarea y .
17. Scrieți un program care citește de la tastatură un număr natural n ($n \in [2, 20]$), apoi n numere naturale din intervalul $[0, 10^4]$, reprezentând, de la stânga la dreapta, în această ordine, valorile elementelor aflate pe prima linie a unui tablou bidimensional cu n linii și n coloane. Programul construiește în memorie tabloul, inițializând celelalte elemente, astfel încât fiecare linie să se obțină prin permutarea circulară a elementelor liniei anterioare, de la stânga spre dreapta, cu o poziție. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele de pe aceeași linie fiind separate prin câte un spațiu. (Examen de Bacalaureat - Iunie 2015)
18. Scrieți un program care citește de la tastatură un număr natural par n ($n \in [2, 50]$) și elementele unui tablou bidimensional cu n linii și n coloane, numere reale, apoi transformă tabloul în memorie, inserând o linie nouă, la mijlocul său, valoarea fiecărui element al acesteia fiind egală cu media aritmetică a elementelor aflate pe coloana corespunzătoare lui, în tabloul citit. Tabloul obținut se afișează pe ecran, câte o linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spațiu. (Examen de Bacalaureat - August 2017)

3.2.4. Diagonale

1. Să se scrie un program care afișează elementele de pe diagonala principală a unei matrice pătratice A cu n linii și n coloane.
2. Să se scrie un program care afișează elementele de pe diagonala secundară a unei matrice pătratice A cu n linii și n coloane.
3. Să se scrie un program care însumează elementele de pe diagonala principală a unei matrice pătratice A cu n linii și n coloane.

4. Să se scrie un program care însumează elementele de pe diagonala secundară a unei matrice pătratice A cu n linii și n coloane.
5. Se dă o matrice cu n linii și n coloane și elemente numere naturale. Calculați diferența în valoare absolută dintre sumele elementelor de pe cele două diagonale.
6. Fie A o matrice pătratică de dimensiune n , cu elemente numere întregi. Să se schimbe între ele elementele care se găsesc pe diagonala principală (se vor inversa elementele simetrice).
7. Fie A o matrice pătratică de dimensiune n , cu elemente numere întregi. Să se schimbe între ele elementele care se găsesc pe diagonala secundară (se vor inversa elementele simetrice).
8. Fie A o matrice pătratică de dimensiune n , cu elemente numere întregi. Să se schimbe între ele elementele care se găsesc pe diagonala principală cu cele care se găsesc pe diagonala secundară (se vor inversa elementele simetrice).
9. Fie A o matrice pătratică cu n linii și n coloane. Să se scrie un program care afișează elementele de deasupra diagonalei principale.
10. Fie A o matrice pătratică cu n linii și n coloane. Să se scrie un program care afișează elementele de sub diagonala principală.
11. Fie A o matrice pătratică cu n linii și n coloane. Să se scrie un program care afișează elementele de deasupra diagonalei secundare.
12. Fie A o matrice pătratică cu n linii și n coloane. Să se scrie un program care afișează elementele de sub diagonala secundară.
13. Fie A o matrice pătratică cu n linii și n coloane. Să se scrie un program care însumează elementele pare de deasupra diagonalei principale.
14. Fie A o matrice pătratică cu n linii și n coloane. Să se scrie un program care însumează elementele impare de sub diagonala principală.
15. Fie A o matrice pătratică cu n linii și n coloane. Să se scrie un program care însumează elementele pozitive de deasupra diagonalei secundare.
16. Fie A o matrice pătratică cu n linii și n coloane. Să se scrie un program care însumează elementele negative de sub diagonala secundară.
17. Se consideră o matrice cu n linii și n coloane cu elemente numere naturale. Afișați divizorii sumei elementelor de deasupra diagonalei principale și ai sumei elementelor de sub diagonala principală.
18. Fiind dată o matrice A cu n linii și n coloane, să se verifice dacă este simetrică față de diagonala principală.
19. Fiind dată o matrice A cu n linii și n coloane, să se verifice dacă este simetrică față de diagonala secundară.
20. Se consideră o matrice cu $n \times n$ elemente numere întregi. Să se determine media aritmetică a elementelor negative din matrice, care sunt situate deasupra diagonalei secundare.

21. Se dă o matrice cu elemente numere întregi. Să se determine media aritmetică a elementelor strict pozitive din matrice, care sunt situate sub diagonala principală. (Variante Bacalaureat 2009)
22. O matrice pătratică cu n linii și n coloane se numește *triunghiulară superior*, dacă toate elementele aflate sub diagonala principală sunt nule. Realizați un program care verifică dacă matricea este triunghiulară superior.
23. O matrice pătratică cu n linii și n coloane se numește *triunghiulară inferior*, dacă toate elementele aflate deasupra diagonalei principale sunt nule. Realizați un program care verifică dacă matricea este triunghiulară superior.
24. Se dă o matrice pătratică cu $n \times n$ elemente. Se consideră că diagonalele împart matricea în 4 zone: *nord*, *sud*, *est* și *vest* (elementele de pe diagonale nu sunt incluse în nici una din zone). Să se calculeze suma elementelor din fiecare din cele 4 zone și să se afișeze.
25. Se dă o matrice cu n linii și n coloane și elemente numere naturale. Să se afișeze, în ordine crescătoare, sumele elementelor din cele patru zone delimitate de diagonale.
26. Se consideră o matrice cu n linii și n coloane și elemente numere naturale. Să se determine suma elementelor de pe cele două diagonale vecine cu diagonala principală (de o parte și de alta a diagonalei).
27. Se consideră o matrice cu n linii și n coloane și elemente numere naturale. Să se determine suma elementelor de pe cele două diagonale vecine cu diagonala secundară (din stânga și din dreapta).
28. Se dă o matrice pătratică cu n linii și n coloane și elemente numere naturale de maxim 3 cifre. Să se afișeze în ordine strict crescătoare, valorile situate sub diagonala principală. Dacă o valoare apare în zona respectivă de mai multe ori, se va afișa o singură dată.
29. Se dă o matrice pătratică cu n linii și n coloane și elemente numere naturale de maxim 3 cifre. Să se afișeze în ordine descrescătoare valorile care apar desupra diagonalei secundare. Fiecare valoare se va afișa o singură dată.
30. Scrieți un program care construiește în memorie o matrice pătratică cu n linii și n coloane formată numai din valori 1 și 2 astfel încât elementele de pe diagonala secundară și cea principală să fie egale cu 1, iar restul elementelor din matrice să fie egale cu 2. Valoarea lui n se citește de la tastatură, iar matricea se va afișa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu câte un spațiu între elementele fiecărei linii. (Variante Bacalaureat 2009)
31. Scrieți un program care citește de la tastatură un număr natural n și construiește în memorie o matrice cu n linii și n coloane ale cărei elemente vor primi valori după cum urmează:
 - elementele aflate pe diagonala principală a matricei vor primi valoarea 0;
 - elementele de pe prima coloană, cu excepția celui aflat pe diagonala principală vor primi valoarea n ;
 - elementele de pe a doua coloană, cu excepția celui aflat pe diagonala principală vor primi valoarea $n-1$
 - ...
 - elementele de pe ultima coloană, cu excepția celui aflat pe diagonala principală vor primi valoarea 1. (Variante Bacalaureat 2009)
32. Scrieți un program care citește de la tastatură un număr natural n și construiește în memorie o matrice cu n linii și n coloane în care elementele de pe cele două diagonale sunt egale cu 0, elementele care se află deasupra ambelor diagonale sunt egale cu 1, elementele care se află sub ambele diagonale sunt egale cu 2, iar restul elementelor sunt egale cu 3. (Variante Bacalaureat 2009)
33. Scrieți un program care citește de la tastatură un număr natural n și construiește în memorie o matrice cu n linii și n coloane în care ultima linie conține, în ordine, numerele $1, 2, 3, \dots, n$, elementele situate deasupra diagonalei principale sunt nule și oricare alt element este obținut prin însumarea elementelor vecine cu el, aflate pe linia imediat următoare, pe aceeași coloană cu el sau pe una din coloanele alăturate. (Variante Bacalaureat 2009)
34. Scrieți un program care citește de la tastatură un număr natural n ($2 < n < 30$) și construiește în memorie o matrice pătratică cu n linii și n coloane formată numai din valori ce aparțin mulțimii $\{1, 2, 3, \dots, n\}$ astfel încât elementele din matrice situate pe diagonala secundară să fie egale cu n , elementele situate pe celelalte două *semidiagonale* paralele cu diagonala secundară și alăturate diagonalei secundare să fie $n-1$, elementele situate pe următoarele două *semidiagonale* paralele cu diagonala secundară de o parte și de alta a acesteia, să fie egale cu $n-2$ etc. Matricea se va afișa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu spații între elementele fiecărei linii. (Variante Bacalaureat 2009)
35. Să se determine *punctele șa* dintr-o matrice A cu m linii și n coloane (elementul a_{ij} este punct șa al matricei A , dacă el este minim pe linia i și maxim pe coloana j sau invers).
36. Se consideră două matrice binare A și B cu n linii și n coloane, fiecare linie reprezentând un număr în baza 2. Să se construiască matricea C , astfel încât linia i să reprezinte cea mai mică, în ordine lexicografică, dintre liniile i din matricele A și B .
37. Se consideră două matrice binare A și B cu n linii și n coloane, fiecare linie reprezentând un număr în baza 2. Să se construiască matricea C , astfel încât linia i să reprezinte suma liniilor i din matricele A și B (ca sumă a două numere în baza 2). În cazul în care se depășește dimensiunea matricei C , aceasta se va suplimenta cu o coloană.

38. Se consideră un tablou bidimensional A cu 200 de linii și 200 de coloane (cu liniile și coloanele numerotate de la 1 la 200), în care, pe fiecare linie toate elementele au valoarea 0, cu excepția celor de pe diagonala principală și a celor aflate imediat în stânga și imediat în dreapta acestora. Elementele nenule sunt completate în ordine, linie cu linie de sus în jos și pe aceeași linie de la stânga la dreapta, cu valorile 1, 2, 3, ..., ca în exemplul

1	2	0	0	0	...	0	0	0
3	4	5	0	0	...	0	0	0
0	6	7	8	0	...	0	0	0
...								
0	0	0	0	0	...	594	595	596
0	0	0	0	0	...	0	597	598

Pentru tabloul descris anterior, se citește un număr natural k ($1 < k < 598$). Se cere să se afișeze două numere naturale reprezentând linia și respectiv coloana pe care se află valoarea k în acest tablou. (Variante Bacalaureat 2009)

3.2.5. Căutări de elemente

- Fiind dată o matrice A cu $m \times n$ elemente numere întregi, să se determine de câte ori apare o valoare dată x ca element al matricei.
- Să se determine dacă un număr se găsește printre elementele diagonalelor unei matrice pătratice cu n linii și n coloane.
- Să se determine dacă un număr x se găsește într-o matrice, în care elementele liniilor sunt sortate crescător și primul element al unei linii este mai mic sau egal decât ultimul element al liniei precedente.
- Să se determine dacă un număr se găsește într-o matrice, în care elementele liniilor sunt sortate descrescător și primul element al unei linii este mai mare sau egal decât ultimul element al liniei următoare.
- Să se determine dacă un număr se găsește într-o matrice, știind că elementele coloanelor sunt sortate crescător și primul element al unei coloane este mai mare sau egal decât ultimul element al coloanei precedente.
- Să se determine dacă un număr se găsește într-o matrice pătratică în care elementele aflate pe liniile paralele cu diagonala principală sunt ordonate crescător, iar oricare element aflat pe o astfel de linie este mai mare sau egal decât toate elementele aflate pe o linie din dreapta sa.
- Să se determine dacă un număr se găsește într-o matrice pătratică în care elementele sunt dispuse în spirală în ordine crescătoare. În caz afirmativ să se afișeze poziția (indicii de linie și de coloană) în care se găsește elementul căutat. În caz contrar se va afișa mesajul *Nu există*.

- Să se determine dacă un număr se găsește pe conturul unei matrice pătratice care conține elementele de la 1 la n^2 , dispuse în ordine crescătoare.
- Să se verifice dacă există o valoare comună tuturor liniilor, respectiv tuturor coloanelor unei matrice A cu m linii și n coloane.
- Se dau două matrice pătratice, matricea A de dimensiune m și matricea B de dimensiune n . Să se spună de câte ori se regăsește matricea B în matricea A .

3.2.6. Verificare proprietate

- Fie A o matrice cu m linii și n coloane cu numere întregi și un vector v cu n componente numere întregi. Să se determine dacă acest vector apare ca linie în matricea A , iar în caz afirmativ să se precizeze numărul acestei linii (acestor linii).
- Se dă o matrice A cu m linii și n coloane cu elemente numere întregi. Să se precizeze dacă matricea conține cel puțin o linie cu elementele în ordine crescătoare, iar în caz afirmativ să se precizeze numărul liniei (numerele liniilor).
- Se dă o matrice A cu m linii și n coloane cu elemente numere întregi. Să se precizeze dacă matricea conține cel puțin o linie care să conțină același element, iar în caz afirmativ să se precizeze numărul liniei (numerele liniilor).
- Se dă o matrice A cu m linii și n coloane cu elemente numere întregi. Să se precizeze dacă matricea conține cel puțin un pătrat perfect. În caz afirmativ se va afișa numărul (numerele) și poziția (pozițiile) pe care se află.
- Fie A o matrice cu m linii și n coloane. Să se verifice dacă matricea dată conține cel puțin două linii identice.
- Fie A o matrice cu m linii și n coloane. Să se verifice dacă matricea dată conține cel puțin două coloane identice.
- Fiind dată o matrice A cu $m \times n$ elemente, să se verifice dacă există o valoare comună tuturor liniilor, respectiv tuturor coloanelor matricei.
- Se dă o matrice cu n linii și m coloane cu elemente numere naturale. Să se determine dacă matricea are cel puțin o linie în care elementele alăturate să aibă parități diferite.
- Scrieți un program care citește de la tastatură două numere naturale m și n ($2 \leq m \leq 50, 2 \leq n \leq 50$) și elementele unui tablou bidimensional cu m linii și n coloane, numere naturale cu cel mult patru cifre fiecare. Programul determină coloanele care au toate elementele egale cu aceeași valoare și, pentru fiecare astfel de coloană afișează pe ecran valoarea respectivă. Valorile afișate sunt separate prin câte un spațiu, iar dacă nu există astfel de valori, programul afișează pe ecran mesajul *Nu există*. (Examen de Bacalaureat - Iunie 2013)

3.2.7. Maxime și minime

1. Fie o matrice A cu m linii și n coloane cu elemente numere întregi. Să se determine elementul maxim al matricei.
2. Fie o matrice A cu m linii și n coloane cu elemente numere întregi. Să se determine elementul minim al matricei, precum și linia, respectiv coloana pe care se găsește în matrice. În cazul în care apare de mai multe ori se vor preciza toate pozițiile.
3. Fiind dată o matrice A cu m linii și n coloane, să se precizeze elementele maxime de pe fiecare linie a matricei.
4. Fiind dată o matrice A cu m linii și n coloane, să se precizeze elementele minime de pe fiecare coloană a matricei.
5. Fie A o matrice cu m linii și n coloane cu elemente numere întregi. Să se afișeze maximul dintre elementele pare ale unei linii L date.
6. Se citește de la tastatură o matrice A cu m linii și n coloane cu elemente numere întregi. Să se determine linia (liniile) din matrice care conține cele mai multe elemente nenule.
7. Se citește de la tastatură o matrice A cu m linii și n coloane cu elemente numere întregi. Să se determine linia (liniile) din matrice care conține cele mai multe elemente distincte.
8. Fie A o matrice cu m linii și n coloane cu elemente de 0 și 1 . Să se scrie un program care afișează numărul maxim de elemente egale cu 1 de pe linii.
9. Se dă o matrice cu $m \times n$ elemente numere întregi. Să se determine cel mai mare număr de pe marginea matricei, precum și de câte ori apare el.
10. Fie A o matrice cu $m \times n$ elemente numere întregi. Să se determine linia l din matrice pe care suma elementelor este maximă.
11. Se dă o matrice cu n linii și m coloane și elemente numere naturale. Să se determine elementul cu număr maxim de apariții în matrice. Dacă există mai multe elemente cu număr maxim de apariții se va afișa cel mai mare dintre ele.
12. Fie A o matrice pătratică de dimensiune n , cu elemente numere întregi. Să se schimbe între ele elementele care se găsesc pe diagonala principală cu cele care se găsesc pe diagonala secundară (se vor inversa elementele simetrice). Să se determine apoi, maximul elementelor matricei, excluzându-le pe cele aflate pe diagonale.
13. Se consideră matricea A de dimensiune n , cu elemente numere întregi. Să se construiască vectorii u și v de dimensiune n , astfel încât x_i să conțină numărul elementelor pare de pe linia i și y_i să conțină numărul elementelor impare de pe linia i . Să se calculeze valoarea $\max(|x_i - y_i|)$, pentru $i = 1, 2, 3, \dots, n$ și să se afișeze.
14. Se citește de la tastatură o matrice a cu m linii și n coloane și elemente numere întregi. Să se scrie un program care localizează în această matrice o zonă dreptunghiulară cu proprietatea că suma elementelor din respectiva zonă (inclusiv laturile) este maximă. În cazul în care sunt mai multe soluții se va afișa oricare dintre ele. Programul va afișa suma astfel rezultată.

15. Se consideră o matrice pătratică a cu n linii și n coloane cu elemente numere întregi. Să se rearanjeze elementele matricei, astfel încât suma elementelor de pe diagonala principală să fie minimă.
16. Scrieți un program care citește de la tastatură un tablou bidimensional cu n linii și n coloane și afișează pe ecran cel mai mare vârf al matricei. (un vârf este un element pentru care toți vecinii lui sunt mai mici decât el).
17. Fiind date două matrice A și B cu m linii și n coloane, să se formeze o matrice C în care elementul de pe fiecare poziție este egal cu maximul dintre elementele aflate pe poziția respectivă în matricele date. Să se afișeze apoi matricea C .
18. Fie A o matrice cu elemente numere reale. Să se formeze un șir X definit în felul următor: x_1 este maximul elementelor de pe coloanele 1 și 2, x_2 este maximul elementelor de pe coloanele 3 și 4 etc. Dacă matricea are un număr impar de coloane, atunci ultimul element al șirului este maximul elementelor de pe ultima coloană a matricei.

3.2.8. Sortări și interclasări

1. Se consideră o matrice A cu $m \times n$ elemente numere întregi. Să se ordoneze crescător elementele de pe prima linie a matricei.
2. Se consideră o matrice A cu $m \times n$ elemente numere întregi. Să se ordoneze crescător elementele de pe ultima coloană a matricei.
3. Se consideră o matrice A cu $m \times n$ elemente numere întregi. Să se ordoneze crescător elementele de pe diagonala principală a matricei (pozițiile celorlalte elemente vor rămâne la fel).
4. Se consideră o matrice A cu $m \times n$ elemente numere întregi. Să se ordoneze crescător elementele de pe diagonala secundară a matricei (pozițiile celorlalte elemente vor rămâne la fel).
5. Se consideră o matrice A cu $m \times n$ elemente numere întregi. Să se schimbe între ele coloanele matricei A , astfel încât elementele de pe ultima linie să fie ordonate crescător.
6. Se consideră o matrice A cu $m \times n$ elemente numere întregi. Să se schimbe între ele liniile matricei A , astfel încât prima coloană să fie ordonată descrescător.
7. Se consideră o matrice pătratică A cu n linii și n coloane cu elemente numere întregi. Să se rearanjeze elementele matricei, astfel încât elementele de pe fiecare linie să fie ordonate crescător.
8. Se consideră o matrice A cu $m \times n$ elemente numere întregi. Să se schimbe între ele coloanele și liniile matricei A , astfel încât elementele de pe diagonala principală să fie ordonate crescător.
9. Se consideră o matrice A cu $m \times n$ elemente numere întregi. Să se schimbe între ele coloanele și liniile matricei A , astfel încât elementele de pe diagonala secundară să fie ordonate descrescător.

1. Se dă o matrice cu n linii și m coloane cu elemente numere naturale. Să se ordoneze liniile matricei crescător după suma elementelor.
2. Se citește de la tastatură două numere naturale n și m ($1 < m < 10, 1 < n < 10$) și o matrice A cu n linii și m coloane formată din numere întregi de cel mult 4 cifre fiecare. Scrieți un program ce sortează descrescător elementele fiecărei linii. Matricea sortată se va afișa pe ecran, fiecare linie a matricei pe câte o linie a ecranului, elementele unei linii fiind separate prin spații. (Variante Bacalaureat 2009)
3. Se consideră un tablou bidimensional cu $n \times m$ elemente numere întregi. Să se creeze un program care ordonează crescător elementele situate pe liniile cu număr de ordine par (linia a doua, linia a patra etc.). La afișare, elementele tabloului vor fi scrise sub forma unui tablou cu n linii și m coloane.
3. Se consideră un tablou bidimensional cu $n \times m$ elemente numere întregi. Să se creeze un program care ordonează descrescător elementele situate pe liniile cu număr de ordine impar (prima linie, linia a treia etc). La afișare, elementele tabloului vor fi scrise sub forma unui tablou cu n linii și m coloane.
14. Se consideră o matrice pătratică A cu n linii și n coloane cu elemente numere întregi. Să se rearanjeze elementele matricei, astfel încât să fie îndeplinite condițiile: elementele de pe fiecare linie să fie ordonate crescător și suma elementelor de pe diagonala principală să fie minimă.
15. Se consideră o matrice pătratică A cu n linii și n coloane cu elemente numere întregi. Să se rearanjeze elementele matricei, astfel încât să fie îndeplinite condițiile: elementele de pe fiecare linie să fie ordonate crescător și suma elementelor de pe diagonala secundară să fie minimă.
16. Se citește de la tastatură o matrice A cu n linii și n coloane cu elemente numere întregi. Să se aranjeze elementele matricei, astfel încât parcurgând matricea în spirală (dinspre exterior spre interior), să se obțină elementele în ordine crescătoare. Nu se vor folosi alte date structurate.
17. Să se aranjeze elementele unei matrice pătratice cu n linii și n coloane cu elemente numere întregi, să se parcurgă matricea pe diagonalele paralele cu diagonala principală, începând cu cea mai de sus diagonală și dinspre stânga spre dreapta să se obțină elementele în ordine crescătoare, ca în exemplul:

1	9	7	20	12	6	2	1
3	128	6	2	25	15	7	3
34	12	33	15	34	26	16	8
26	8	16	45	128	45	33	20

3.3. Șiruri de caractere

3.3.1. Parcurgeri

1. Se consideră un șir de caractere. Să se determine câte vocale conține șirul.
2. Se consideră un șir de caractere. Să se determine numărul de majuscule din șir.
3. Se consideră un șir de caractere. Să se determine numărul de cifre din șir.
4. Se consideră un șir de caractere. Să se determine numărul de caractere speciale din șir.
5. Se consideră un șir de caractere. Să se scrie un program care va afișa doar cifrele din șir.
6. Se consideră un șir de caractere. Să se scrie un program care va afișa doar consoanele din șir.
7. Se citește de la tastatură o frază cu maxim 250 de caractere. Se cere să se afișeze câte semne de punctuație sunt. Semne de punctuație se consideră următoarele caractere: { . , ; - ? ! }
8. Se dă un șir de caractere. Să se determine câte vocale din șir sunt cuprinse între două consoane.
9. Se citește de la tastatură un șir de caractere care conține și cifre. Scrieți un program care să determine dacă unul dintre numerele care se pot forma cu cifrele din șirul de caractere este par.
10. Să se scrie un program care citește un șir de caractere format din litere mici ale alfabetului englez și transformă fiecare vocală în literă mare.
11. Să se verifice dacă un cuvânt dat este palindrom (citit de la dreapta la stânga sau de la stânga la dreapta reprezintă același cuvânt).
12. Să se scrie un program care să afișeze prefixele unui cuvânt citit.
13. Să se scrie un program care să afișeze sufixele unui cuvânt citit.
14. Se citește de la tastatură două șiruri de caractere formate numai din cifre. Se cere să se calculeze suma și diferența dintre cele două numere obținute din cele 2 șiruri de caractere citite.
15. Să se scrie un program care citește o propoziție și determină numărul de cuvinte care încep și se termină cu vocală. (Variante Bacalaureat 2009)
16. Se citește de la tastatură un șir de caractere care conține numai litere și cifre. Să se afișeze numărul literelor mari, numărul literelor mici și numărul caracterelor cifră din text.
17. Scrieți un program care citește de la tastatură un șir de caractere format din cel mult 20 de litere mici din alfabetul englez și care afișează mesajul DA, dacă șirul este format dintr-un număr egal de consoane și vocale (a, e, i, o, u), respectiv mesajul NU în caz contrar.
18. Se consideră un cuvânt format din cel puțin două și cel mult 100 de caractere, numai litere mici ale alfabetului englez. Scrieți un program care citește de la tastatură un cuvânt de tipul precizat și afișează pe ecran mesajul DA în cazul în care cuvântul conține doar consoane și, eventual, vocala i , sau mesajul NU în caz contrar. (Examen de Bacalaureat - August 2015)

9. Să se scrie un program care citește un șir de caractere și afișează litera mică cel mai des întâlnită în șir. (Variante Bacalaureat 2009)
10. Scrieți un program care convertește un șir de caractere într-un întreg. Conversia se va opri la primul caracter care nu este cifră.
11. Scrieți un program care citește de la tastatură un șir de cel mult 50 de caractere (litere mici și mari ale alfabetului englez, cifre, puncte, virgule și spații) și afișează pe ecran cifra care apare de cele mai multe ori în șirul citit. (Variante Bacalaureat 2009)
12. Se dau 2 șiruri de caractere. Să se afișeze caracterele comune celor două șiruri.
13. Se citește de la tastatură un șir de caractere. Să se rearanjeze caracterele, astfel încât vocalele să se situeze pe primele poziții în șir.
14. Spunem despre un șir de caractere că are perioada k , dacă șirul este format prin concatenarea de mai multe ori a unui șir de lungime k . Să se determine cea mai scurtă perioadă pentru un șir dat, precum și subșirul corespunzător acesteia.
15. Să se scrie un program care determină cel mai bun șablon comun a două șiruri de caractere, adică șablonul comun cu număr minim de caractere (Un șir de caractere s este șablon pentru un șir de caractere x dacă are aceeași lungime cu x și este alcătuit numai din caractere ale mulțimii $\{*, #, ?\}$, iar pe fiecare poziție din s în care apare $*$ în x apare vocală, pe fiecare poziție din s în care apare $#$ în x apare consoană, iar pe fiecare poziție din s în care apare $?$ în x putem avea orice caracter. (Variante Bacalaureat 2009)
26. Să se scrie un program care citește un șir de caractere și afișează litera mică cel mai des întâlnită în șir. (Variante Bacalaureat 2009)
27. Se citește de la tastatură un șir de maxim 100 de caractere, litere mici ale alfabetului englez. Să se găsească cea mai lungă secvență care conține litere ordonate alfabetic.
28. Scrieți un program care citește de la tastatură două șiruri de caractere, fiecare șir fiind format din cel mult 100 de litere mici din alfabetul englez, și care afișează mesajul DA dacă toate literele din primul șir se găsesc în cel de-al doilea șir, nu neapărat în aceeași ordine și de același număr de ori, sau mesajul NU în caz contrar.
29. Se consideră un șir de caractere de maxim 100 de caractere, format din litere mici ale alfabetului englez. Se numește *bâlbă*, o secvență de caractere care apare de cel puțin două ori una după alta. Să se determine cea mai lungă bâlbă din șirul dat.
30. Se citește de la tastatură un șir de caractere, reprezentând un număr întreg scris în sistemul de numerație roman. Să se scrie un program care determină scrierea arabă a numărului respectiv. Cifrele din sistemul roman sunt:

Cifra din sistemul roman

I
V
X
L
C
D
M

Valoarea din sistemul arab

1
5
10
50
100
500
1000

Pentru reprezentarea numerelor se folosesc următoarele reguli:

- Mai multe cifre identice dintr-o succesiune formează un număr egal cu suma cifrelor
- Un grup de două cifre, dintre care prima este mai mică decât a doua reprezintă un număr care are valoarea egală cu diferența dintre valoarea cifrei mai mică și valoarea cifrei mai mare (*Exemplu IV* – reprezintă 4)
- Un grup de două cifre în care prima are o valoare mai mare decât a doua, reprezintă un număr care are o valoare egală cu suma dintre valoarea primei cifre și valoarea celei de a doua (*Exemplu VII* – reprezintă 7) (*Exemplu* – Numărul roman CMLXVIII reprezintă numărul arab 968.)

3.3.2. Operații cu șiruri de caractere

1. Se consideră un șir de caractere. Să se scrie un program care să înlocuiască cifrele din șir cu semnul #.
2. Se consideră un șir de caractere. Să se scrie un program care să înlocuiască vocalele din șir cu semnul \$.
3. Să se scrie un program care citește un cuvânt format din litere mici ale alfabetului englez și elimină din cuvânt toate vocalele.
4. Să se scrie un program care citește un cuvânt format din litere mici ale alfabetului englez și elimină din cuvânt toate caracterele cifră. Să se precizeze lungimea anterioară și cea curentă a șirului.
5. Să se scrie un program care citește un cuvânt și inserează după fiecare vocală caracterul #.
6. Să se scrie un program care citește un cuvânt și inserează înainte de fiecare vocală caracterul %.
7. *Limba păsărească* este o limbă pe care o folosesc copiii în joacă! Un text scris în română se traduce în păsărește astfel: după fiecare vocală se inserează litera *p* și vocala respectivă. Dându-se o propoziție scrisă în limba română, să se traducă în limba păsărească.
8. Să se elimine toate aparițiile caracterului spațiu dintr-un text.
9. Să se scrie un program care citește de la tastatură un cuvânt de maximum 20 de litere și minimum o literă și șterge litera din mijloc în cazul în care cuvântul are un număr impar de litere, respectiv cele două litere din mijlocul cuvântului dacă acesta are un număr par de litere. Programul va afișa cuvântul rezultat în urma ștergerii sau mesajul CUVANT VID dacă după ștergere acesta rămâne fără nici o literă.

10. Scrieți un program care citește de la tastatură un cuvânt de cel mult 15 litere mici ale alfabetului englez și afișează pe ecran, pe linii distincte, cuvintele obținute prin ștergerea succesivă a vocalelor din cuvântul citit, de la stânga la dreapta.
11. Se citește de la tastatură un cuvânt de cel mult 20 de caractere, format din litere mici ale alfabetului englez. Să se afișeze toate cuvintele distincte ce se pot forma prin eliminarea câte unui singur caracter din cuvântul dat.
12. Scrieți un program care citește de la tastatură un cuvânt de cel mult 10 litere mici ale alfabetului englez și scrie pe ecran, pe linii distincte, cuvintele obținute prin ștergerea succesivă a vocalelor în ordinea alfabetică a lor (a, e, i, o, u). La fiecare pas se vor șterge toate aparițiile din cuvânt ale unei vocale.
13. Se dau două șiruri de caractere s și t . Să se elimine din s prima apariție a lui t .
14. Se consideră un text cu cel mult 100 de caractere (litere mici ale alfabetului englez și spații), în care cuvintele sunt separate prin unul sau mai multe spații. Scrieți un program care citește de la tastatură un text de tipul menționat mai sus și determină transformarea acestuia în memorie prin eliminarea unor vocale, astfel încât în nici un cuvânt din text să nu existe două vocale pe poziții alăturate. Programul afișează pe ecran textul obținut. Se consideră vocale literele a, e, i, o, u . (Examen de Bacalaureat - Model 2014)
15. Scrieți un program care citește de la tastatură un șir de cel mult 50 de caractere (cifre, litere ale alfabetului englez și spații; șirul conține cel puțin o literă), apoi construiește în memorie și afișează pe ecran șirul de caractere obținut din șirul citit prin eliminarea tuturor caracterelor care nu sunt litere. (Variante Bacalaureat 2009)
16. Să se scrie un program care citește un șir de caractere format din litere mici ale alfabetului englez și elimină din șir toate consoanele.
17. Se consideră un cuvânt format din cel puțin două și cel mult 100 de caractere, numai litere mici ale alfabetului englez. Scrieți un program care citește de la tastatură un cuvânt de tipul precizat și îl transformă în memorie, eliminând un număr minim de litere, astfel încât el să conțină doar vocale și, eventual, consoana m . Programul afișează pe ecran cuvântul obținut, sau mesajul *Nu există* dacă nu se poate obține un astfel de cuvânt. (Examen de Bacalaureat - Model 2016)
18. Se dau două șiruri de caractere s_1 și s_2 . Să se elimine din s_1 toate aparițiile lui s_2 .
19. Să se scrie un program care citește de la tastatură un șir de caractere și elimină din șir toate perechile de vocale consecutive. (Variante Bacalaureat 2009)
20. Se citește un text de cel mult 255 caractere și două litere c_1 și c_2 . Înlocuiți în textul citit toate aparițiile literei c_1 cu litera c_2 .

21. Se citește un text de cel mult 255 caractere. Inversați în textul citit caracterele simetrice.
22. Scrieți un program care citește de la tastatură un cuvânt format din cel mult 20 de caractere, doar litere ale alfabetului englez (cel puțin o majusculă și cel puțin o literă mică). Programul determină transformarea cuvântului citit prin eliminarea fiecărei litere mici a cuvântului, restul literelor nemodificându-se. Programul afișează pe ecran cuvântul obținut.
23. Se citesc două cuvinte de la tastatură. Să se scrie un program care să verifice dacă cele două cuvinte se obțin unul din celălalt prin permutarea literelor.
24. Se consideră un text cu maximum 255 de caractere, format din litere mici ale alfabetului englez și spații. Textul conține cel puțin o consoană. Scrieți un program care citește de la tastatură textul și apoi determină transformarea acestuia, eliminând numai ultima consoană care apare în text. Programul va afișa pe ecran textul obținut.
25. Se citește un cuvânt format din litere mici. Să se afișeze cuvintele care se obțin prin eliminarea succesivă a primei și ultimei litere din șirul citit.
26. Pentru compactarea unor mesaje se înlocuiește orice succesiune de mai multe caractere de același fel cu un caracter de control ('#'), urmat de caracterul respectiv și de numărul de repetări succesive ale caracterului. Să se scrie programul de compactare și decompactare a unui text citit de la tastatură.
27. Scrieți un program care citește de la tastatură un cuvânt s de cel mult 20 de litere mici ale alfabetului englez, construiește în memorie și afișează pe ecran cuvântul s după eliminarea primei și a ultimei vocale. Cuvântul s conține cel puțin două vocale. Se consideră vocale literele: a, e, i, o, u .
28. Scrieți un program care citește de la tastatură un șir de cel mult 100 de caractere, format doar din literele mici ale alfabetului englez și caractere spațiu. Programul afișează pe ecran șirul codificat astfel: după fiecare vocală se adaugă consoana imediat următoare (după a se inserează b , după i se inserează j etc.). La sfârșitul șirului se adaugă cuvântul *STOP*.
29. Se consideră un text cu cel mult 100 de caractere (litere mici ale alfabetului englez și spații), care conține cel puțin o vocală. Scrieți un program care citește de la tastatură un text de tipul menționat mai sus și determină transformarea acestuia în memorie prin duplicarea tuturor vocalelor din text. Programul afișează pe ecran textul obținut. Se consideră vocale literele a, e, i, o, u . (Examen de bacalaureat - Simulare 2014)
30. Se consideră două șiruri de caractere a , de lungime na și b , de lungime nb . Șirul a este numit *prefix* al șirului b dacă $na \leq nb$ și subșirul lui b determinat de primele sale na caractere coincide cu a . Șirul a este numit *suffix* al șirului b dacă $na \leq nb$ și subșirul lui b determinat de ultimele sale na caractere coincide cu a . Scrieți un program care citește de la tastatură două cuvinte, fiecare fiind format din cel mult 30 de caractere, numai litere

mici ale alfabetului englez. După primul cuvânt se tastează Enter. Programul afișează pe ecran, separate prin câte un spațiu, toate sufixele primului cuvânt care sunt prefixe ale celui de al doilea cuvânt. Dacă nu există astfel de subșiruri, se afișează pe ecran mesajul *NU EXISTĂ*. (Examen de Bacalaureat - Iunie 2012)

31. Scrieți un program care citește de la tastatură două cuvinte distincte, fiecare fiind format din cel mult 30 de caractere, numai litere mici ale alfabetului englez. După primul cuvânt se tastează Enter. Programul verifică dacă prin eliminarea unor litere din cel de-al doilea cuvânt se poate obține primul cuvânt. În caz afirmativ programul afișează pe ecran mesajul *DA*, altfel mesajul *NU*. (Examen de Bacalaureat – Sesiunea Specială 2012)

3.3.3. Verificare proprietate

1. Să se verifice dacă un cuvânt dat este format numai din litere mici.
2. Să se verifice dacă un șir de caractere conține numai caractere cifră.
3. Să se scrie un program care verifică dacă două cuvinte date sunt *anagrame* (conțin aceleași litere).
4. Să se verifice dacă o propoziție dată este *palindromică*.
5. Să se verifice dacă două cuvinte introduse de la tastatură sunt *rime*. Două cuvinte sunt rime, dacă ultimele x caractere coincid (x este dat de la tastatură).

3.3.4. Construcții cuvinte, șiruri de caractere

1. Să se scrie un program care citește de la tastatură un text și construiește un șir care va conține doar vocalele din textul citit.
2. Se citesc de la tastatură patru cuvinte. Să se scrie un program care va forma un nou cuvânt din cuvintele citite, astfel: din primul cuvânt va adăuga un caracter, din al doilea două caractere, din al treilea trei caractere, iar din al patrulea $n/2$ caractere, unde n este lungimea celui de-al patrulea cuvânt.
3. Fie s și t două șiruri de lungimi egale. Șirul s conține litere mici, iar șirul t conține numai cifre. Se cere să se construiască al treilea șir u , care să conțină literele din s repetate de un număr de ori egal cu cifra corespunzătoare din al doilea șir.
4. Să se scrie un program care citește de la tastatură numele și prenumele unei persoane și construiește un al treilea șir de caractere, care va conține consoanele din prenumele citit dispuse în ordinea în care apar în prenume, urmate de exact un spațiu și de numele citit. (Variante Bacalaureat 2009)
5. Se dă un șir de caractere. Afișați șirul format doar din caracterele aflate pe poziții impare și șirul format din caracterele aflate pe poziții pare. Cele două șiruri vor fi separate printr-un singur spațiu, pe câte o linie.
6. Pentru două șiruri de caractere cu maximum 250 de caractere fiecare (litere mici ale alfabetului englez), cu caracterele în ordine alfabetică, introduse de la tastatură, se cere să se afișeze pe ecran un al treilea șir, format din toate caracterele primelor două, așezate în ordine alfabetică.

7. Se citesc de la tastatură două șiruri de caractere formate din cel mult 50 de litere fiecare. Să se interclaseze cele două șiruri. Șirul format se obține prin preluarea, din fiecare șir, a câte unei litere (prima literă a primului șir, apoi prima literă a celui de-al doilea, apoi a doua literă a primului șir, apoi a doua literă a celui de-al doilea șir etc). Când se termină literele din unul dintre șiruri, se vor prelua toate literele rămase din celălalt șir.

1. Scrieți un program care citește de la tastatură un șir format din cel mult 50 de caractere care reprezintă numele și prenumele unei persoane. Între nume și prenume se află un număr de caractere spațiu. Să se scrie un program care să șteargă spațiile inutile dintre nume și prenume.

1. Scrieți un program care citește de la tastatură un șir, format din cel mult 50 de caractere, care reprezintă numele și prenumele unei persoane. Între nume și prenume se află un număr de caractere spațiu. Să se scrie un program care afișează pe ecran inițiala prenumelui (prima literă a prenumelui), urmată de un caracter punct, de un spațiu și de numele din șirul citit inițial.

3.3.5. Cuvinte și fraze

1. Se citesc de la tastatură un cuvânt de maxim 10 caractere și un număr natural n . Să se multiplice cuvântul de n ori și să se afișeze rezultatul obținut.
2. Se citește un text de cel mult 255 de caractere, format din mai multe cuvinte, separate prin spații. Eliminați din fiecare cuvânt care începe cu o vocală, consoanele. Cuvintele rezultate vor fi trecute în ordinea în care au apărut în fraza inițială, separate de câte un spațiu.
3. Se citește de la tastatură o frază cu maxim 100 de caractere. Să se scrie un program care transformă fiecare literă de la începutul unei propoziții în literă mare.
4. Se consideră un text cu cel mult 70 de caractere (litere mici ale alfabetului englez și spații), în care cuvintele sunt separate prin unul sau mai multe spații. Înaintea primului cuvânt și după ultimul cuvânt nu există spații. Scrieți un program care citește de la tastatură un text de tipul menționat mai sus și afișează pe ecran numărul de cuvinte în care apare litera a . (Examen de Bacalaureat - Model 2013)
5. Să se scrie un program care citește mai multe propoziții și determină propoziția cu cele mai multe vocale.
6. Se citește de la tastatură o frază de maxim 100 de caractere. Între cuvintele frazei pot apărea ca separatori spațiul, virgula și punctul. Să se afișeze cuvintele frazei unul sub altul în ordine alfabetică.
7. Scrieți un program care citește de la tastatură, în această ordine, un număr natural n , apoi n cuvinte, separate prin Enter, urmate de un număr natural k . Numerele citite sunt din intervalul $[1,20]$, iar fiecare cuvânt este format

- din cel mult 20 de caractere, numai litere mici ale alfabetului englez. Cel puțin unul dintre cuvinte are k litere. Programul construiește în memorie, apoi afișează pe ecran, un șir care să cuprindă cuvintele citite, separate prin câte un spațiu, astfel încât toate cele care au k litere să ocupe primele poziții, iar celelalte să se regăsească în continuarea acestora, într-o ordine oarecare. (Examen de Bacalaureat – Sesiunea Specială 2016)
8. Se citește de la tastatură n propoziții formate din caractere ale alfabetului englez. Ultimele caractere ale fiecărei propoziții sunt cifre care formează numere întregi. Să se afișeze aceste propoziții în ordinea crescătoare a numerelor cu care se termină. La afișare nu se vor trece și numerele.
 9. Se citește de la tastatură o frază cu maxim 100 de caractere. Se cere să se afișeze textul obținut prin oglindirea fiecărui cuvânt.
 10. Se consideră un text cu cel mult 100 de caractere, în care cuvintele sunt formate numai din litere mici ale alfabetului englez și sunt separate prin câte un spațiu.
Scrieți un program care citește de la tastatură un text de tipul menționat mai sus și determină transformarea acestuia în memorie, astfel încât din fiecare cuvânt format dintr-un număr impar de litere (cel puțin trei) să se elimine litera aflată pe poziția din mijloc. Programul afișează pe ecran textul obținut, iar dacă nu există nici un cuvânt modificat, afișează pe ecran mesajul *Nu există*. (Examen de Bacalaureat – Sesiunea Specială 2014)
 11. Se citește de la tastatură o frază de maxim 100 de caractere. Între cuvintele frazei pot fi unul sau mai multe spații. Să se afișeze cuvântul (cuvintele) de lungime maximă.
 12. Se citește de la tastatură o frază cu maxim 100 de caractere. Să se scrie un program care afișează frecvențele lungimii cuvintelor care apar în text. Cuvintele frazei sunt separate prin unul sau mai multe spații.
 13. Scrieți un program care citește de la tastatură un număr natural n ($2 \leq n \leq 20$) și apoi n cuvinte distincte, fiecare fiind format din cel mult 20 de caractere, numai litere mici ale alfabetului englez. La introducerea datelor, după fiecare cuvânt se tastează Enter. Programul afișează pe ecran numărul de cuvinte dintre ultimele $n - 1$ citite, care se încheie cu primul cuvânt citit. (Examen de bacalaureat - Model 2012)
 14. Un text are cel mult 100 de caractere, iar cuvintele sale sunt formate doar din litere mici ale alfabetului englez și sunt separate prin câte un spațiu. Scrieți un program care citește de la tastatură un text de tipul precizat mai sus și îl transformă în memorie prin înlocuirea fiecărui cuvânt format din număr par de litere, cu simbolul #. Programul afișează pe ecran textul obținut sau mesajul *Nu există* dacă textul citit nu conține astfel de cuvinte. (Examen de Bacalaureat - Model 2017)
 15. Se consideră un text cu cel mult 100 de caractere (litere mici ale alfabetului englez și spații), în care cuvintele sunt separate prin câte un spațiu. Înaintea primului cuvânt și după ultimul cuvânt nu există spații. Scrieți un program

care citește de la tastatură un text de tipul menționat mai sus și determină transformarea acestuia în memorie prin înlocuirea fiecărui cuvânt format din trei litere cu simbolul *. Programul afișează pe ecran textul obținut. (Examen de Bacalaureat – Sesiunea Specială 2013)

16. Se citește de la tastatură un text format din cuvinte separate între ele prin câte un spațiu. Fiecare cuvânt are cel mult 40 de caractere, doar litere mici ale alfabetului englez. Textul are cel mult 200 de caractere. Scrieți un program care afișează pe ecran, pe linii separate, doar cuvintele din textul citit care conțin cel mult trei vocale. Se consideră vocale: a, e, i, o, u .
17. Se dă o propoziție formată din litere mari și mici ale alfabetului englez și spații. Să se afișeze în ordine alfabetică cuvintele care apar în propoziție.
18. Se dau două propoziții formate din litere mari și mici ale alfabetului englez și spații. Să se afișeze în ordine alfabetică cuvintele care apar în ambele șiruri.
19. Se dă o propoziție formată din litere mici ale alfabetului englez, cuvintele sunt separate prin spații. Să se verifice dacă propoziția dată conține cel puțin un cuvânt palindrom.
20. Se citește de la tastatură o frază de maxim 100 de caractere. Să se afișeze una sub alta, toate perechile de vocale consecutive din frază.
21. Să se scrie un program care citește de la tastatură o propoziție formată din mai multe cuvinte separate prin spații și transformă prima și ultima literă a fiecărui cuvânt în literă mare. (Variante Bacalaureat 2009).
22. Se consideră un text cu cel mult 100 de caractere (litere mici ale alfabetului englez și spații), în care cuvintele sunt separate prin unul sau mai multe spații. Înaintea primului cuvânt și după ultimul cuvânt nu există spațiu. Scrieți un program care citește de la tastatură un text de tipul menționat mai sus și determină transformarea acestuia în memorie prin eliminarea unor spații, astfel încât între oricare două cuvinte alăturate să rămână exact un spațiu. Programul afișează pe ecran textul obținut. (Examen de bacalaureat - August 2013)
23. Se citește de la tastatură o frază cu maxim 100 de caractere. Cuvintele frazei sunt separate prin unul sau mai multe spații. Să se determine numărul cuvintelor din frază care conțin doar vocale.
24. Să se scrie un program care citește mai multe propoziții și determină propoziția cea mai lungă (cu cele mai multe cuvinte).
25. Scrieți un program care citește de la tastatură, în această ordine, un număr natural n , apoi n cuvinte, urmate de un număr natural k . Numerele citite sunt din intervalul $[1, 20]$, iar fiecare cuvânt este format din cel mult 20 de caractere, numai litere mici ale alfabetului englez. Cel puțin unul dintre cuvinte are k litere. Programul construiește în memorie, apoi afișează pe ecran, un șir care să cuprindă cuvintele citite, separate prin câte un spațiu, astfel încât toate cele care au k litere să ocupe primele poziții, iar celelalte să se regăsească în continuarea acestora, într-o ordine oarecare. (Examen de Bacalaureat - Sesiunea specială 2016)

26. Se dă un șir de cuvinte. Să se determine numărul cuvintelor care conțin în componența lor și litere mari.
27. Să se scrie un program care citește de la tastatură o frază. Între cuvintele frazei pot apărea ca separatori unul sau mai multe spații. Să se elimine din frază spațiile inutile.
28. Să se scrie un program care citește un șir de caractere format din cuvinte separate prin spații sau virgule și elimină din șir toate virgulele.
29. Se introduce de la tastatură o frază de cel mult 100 de caractere. Cuvintele frazei pot fi separate prin unul sau mai multe spații. Să se afișeze toate cuvintele din text formate din p caractere.
30. Se introduce de la tastatură o frază de cel mult 100 de caractere. Cuvintele frazei pot fi separate prin unul sau mai multe spații. Să se elimine din frază toate cuvintele din text formate din p caractere. Fraza inițială conține cel puțin un cuvânt care să aibă p caractere.
31. Se consideră un text cu cel mult 100 de caractere, în care cuvintele sunt formate numai din litere mari și mici ale alfabetului englez și sunt separate prin câte un spațiu. Textul reprezintă numele unei instituții sau al unei organizații. Scrieți un program care citește de la tastatură un text de tipul precizat și construiește în memorie, apoi afișează pe ecran, un șir de caractere ce reprezintă *acronimul* corespunzător numelui citit (*acronimul* este format din primul caracter al fiecărui cuvânt al numelui care începe cu majusculă). (Examen de Bacalaureat – Sesiunea Specială 2015)
32. Se consideră un text cu cel mult 100 de caractere, în care cuvintele sunt formate din litere mari ale alfabetului englez și sunt separate prin câte un spațiu. Scrieți un program care citește de la tastatură un text de tipul menționat mai sus și afișează pe ecran, pe câte un rând, cuvintele care cuprind în componența lor cel puțin o notă muzicală. Dacă nu există astfel de cuvinte, se afișează pe ecran mesajul *Nu există*. Notele muzicale sunt: *DO, RE, MI, FA, SOL, LA, SI*. (Examen de Bacalaureat - Sesiunea specială 2017)
33. Se consideră un text alcătuit din cel mult 250 de caractere, în care cuvintele sunt formate doar din litere mici ale alfabetului englez și sunt separate prin unul sau mai multe caractere *. Scrieți un program care citește de la tastatură textul și afișează pe ecran, pe câte o linie, toate secvențele formate din câte două litere identice.
34. Se consideră un text cu cel mult 100 de caractere, format din cuvinte și numere, separate prin unul sau mai multe spații. Cuvintele sunt formate numai din litere ale alfabetului englez. Toate numerele sunt reale, pozitive și sunt formate numai din parte întreagă sau din parte întreagă și parte fracționară, separate prin punctul zecimal (.). Scrieți un program care citește de la tastatură textul, pe care îl transformă, eliminând din componența sa toate numerele care au parte fracționară. Programul afișează apoi pe ecran textul obținut. (Examen de Bacalaureat – August 2010)

3.4. Tipul multime

1. Se citesc n cuvinte. Să se afișeze literele distincte din fiecare cuvânt și din toate cele n cuvinte.
2. Se citește de la tastatură un număr natural n , apoi n numere naturale. Să se afișeze numerele distincte dintre cele citite.
3. Se citesc de la tastatură 2 mulțimi A și B cu câte m , respectiv n elemente numere întregi mai mici decât 255. Să se verifice dacă cele două mulțimi conțin aceleași elemente.
4. Să se afișeze cel mai mare număr obținut din cifrele unui număr n dat, fără duplicaturi.
5. Se dau de la tastatură două numere naturale. Să se determine cifrele comune ale celor două numere.
6. Se citesc de la tastatură n numere naturale. Să se determine cifrele comune celor n numere citite.
7. Să se determine numerele prime mai mici ca 255, folosind tipul multime.
8. Se citesc de la tastatură elementele a 2 mulțimi A și B . Să se determine dacă cele două mulțimi sunt disjuncte (nu au nici un element comun).
9. Se citesc de la tastatură elementele a n mulțimi. Să se scrie un program care verifică dacă mulțimile sunt disjuncte două câte două.
10. Se citesc de la tastatură n cuvinte formate din litere mici ale alfabetului englez. Să se afișeze perechile de cuvinte dintre cele citite, care folosesc aceleași litere în scrierea lor.
11. Se dau două numere x și y mai mici ca 255. Să se scrie un program care determină divizorii comuni ai celor două numere date.

3.5. Tipul înregistrare

1. La un concurs au participat n elevi, pentru fiecare elev cunoscându-se numele, *liceul de proveniență* (șir de caractere) și *media* obținută (de la 1 la 10). Să se realizeze un program care citește datele elevilor și afișează pe ecran numele elevilor cu cea mai mare notă și media notelor participanților.
2. Se citește de la tastatură data nașterii unei persoane. Să se stabilească ce vârstă are persoana.
3. Să se definească o structură care să memoreze lungimea și lățimea unui dreptunghi. Să se scrie un program care citește de la tastatură lungimile și lățimile a n dreptunghiuri, apoi afișează pe ecran dreptunghiul (numărul de ordine) cu aria cea mai mică și dreptunghiul cu diagonala cea mai mare.
4. Se citesc de la tastatură n numere complexe sub forma parte reală și parte imaginară. Să se calculeze suma celor n numere.

5. Se citesc de la tastatură n perechi de numere întregi, reprezentând coordonatele (abscisa și ordonata) a n puncte din plan. Să se afișeze care dintre acestea se află pe dreapta de ecuație $5x + y - 2 = 0$.
6. Se definește un cerc pentru care se cunosc dimensiunea razei și coordonatele centrului cercului. Se citesc de la tastatură coordonatele carteziene a n puncte din plan. Să se scrie un program care afișează câte dintre cele n puncte se află în exteriorul cercului.
7. Să se descompună un număr natural în factori primi, memorând rezultatul sub forma unui vector de înregistrări: fiecare înregistrare va cuprinde două câmpuri, unul indicând factorul prim, iar celălalt puterea la care apare în descompunere.
8. Se consideră un vector cu înregistrări ce conține informații referitoare la elevii unei școli. O înregistrare conține câmpurile nume, clasă, medie. Să se scrie un program care ordonează elevii descrescător după medii, apoi afișează clasa cu cea mai mare medie generală.
9. Se descompune un număr natural în factori primi. Să se afișeze factorii primi care apar în descompunere, împreună cu puterile corespunzătoare lor. Se va folosi un tip înregistrare care va fi format din două câmpuri, unul va conține factorul prim, celălalt va conține puterea corespunzătoare lui din descompunere.
10. Să se descompună un număr natural în factori primi, folosind un vector de înregistrări. Fiecare înregistrare conține două câmpuri, unul pentru numărul care urmează a fi împărțit, iar celălalt, pentru numărul la care se împarte.
11. Pentru evidența medicamentelor dintr-un depozit sunt necesare următoarele informații: *codul medicamentului* (număr întreg), *denumirea* (șir de caractere) și *prețul medicamentului* (număr întreg). Presupunând că în depozit există n medicamente și că se cunosc datele pentru toate, se cere să se testeze dacă medicamentul cu denumirea x (x este dat) se află în depozit, afișându-se un mesaj corespunzător. În caz afirmativ se va scădea prețul cu un procent $p\%$ (p dat) și se va afișa noul preț.
12. Să se scrie un program pentru datele unui concurs de informatică. Câmpurile înregistrărilor vor fi *Nume* (șir de caractere) și *Medie* (număr real). Programul va conține un meniu cu opțiunile:
 - Inițializează vectorul de înregistrări;
 - Adaugă un elev;
 - Elimină un elev cu un nume introdus de la tastatură;
 - Elimină un elev de pe o poziție dată;
 - Calculează mediile;
 - Listează alfabetic elevii participanți la concurs;
 - Listează în ordinea descrescătoare a mediilor participării la concurs;
 - Listează toți elevii cu media mai mare decât o medie m dată.

13. Să se scrie un program pentru gestionarea vânzărilor de produse, știind că pentru fiecare produs se înregistrează: *Denumirea* și *Prețul produsului*, *Cantitatea vândută* și *Valoarea vânzării* produsului respectiv ($\text{Cantitate} \times \text{Preț}$). Programul va conține un meniu din care se poate selecta în mod repetat, atâta timp cât utilizatorul dorește acest lucru, una din următoarele acțiuni: introducerea datelor aferente vânzării unui produs, determinarea valorii medii a tuturor produselor vândute și afișarea informațiilor despre toate vânzările.

3.6. Fișiere

1. Să se afișeze numerele de pe fiecare linie dintr-un fișier text.
2. Să se afișeze numărul de numere de pe fiecare linie a unui fișier text.
3. Fișierul "*Date.in*" conține n numere separate printr-un spațiu. Să se scrie un program care va afișa aceste numere unul sub altul, trecând în dreptul fiecăruia unul dintre cuvintele *PAR* sau *IMPAR*. Rezultatul va fi afișat în fișierul "*Date.out*".
4. Se citește un fișier text care conține numere întregi despărțite prin spațiu. Să se afișeze minimul și maximul numerelor din acel fișier.
5. Fișierul "*Numere.txt*" conține pe prima linie un număr natural n și pe următoarea linie n numere întregi. Să se calculeze și să se afișeze suma elementelor pozitive ale șirului.
6. Se dă fișierul text "*Numere.txt*" care conține pe primul rând o valoare naturală nenulă n și pe al doilea rând n valori întregi de cel mult 4 cifre separate prin unul sau mai multe spații. Se cere să se calculeze suma elementelor pătrate perfecte de pe al doilea rând din fișier.
7. Fișierul "*Date.in*" conține un șir de numere întregi, toate pe un singur rând, separate prin spații. Să se copieze în fișierul text "*Date.out*" elementele pare ale primului șir.
8. Fișierul "*Date.in*" conține un șir de numere întregi, toate pe un singur rând, separate prin spații. Să se copieze în fișierul text "*Date.out*" elementele numere prime ale primului șir.
9. Să se scrie un program care citește din fișierul "*Numere.in*" mai multe numere întregi și afișează pe ecran câte dintre numerele citite sunt pare și câte sunt impare.
10. Fișierul "*Numere.in*" conține pe prima linie a fișierului un număr natural și pe următoarea linie n numere întregi. Să se afișeze în fișierul "*Numere.out*" numerele palindrom conținute de primul fișier.

11. Fișierul *Numere.in* conține pe prima linie un număr natural n ($n \in [2, 10^9]$), iar pe a doua linie un șir de cel mult 10^9 numere naturale din intervalul $[1, n]$. Numerele din șir sunt ordonate crescător și sunt separate prin câte un spațiu. Se cere să se determine valorile naturale distincte din intervalul $[1, n]$ care NU se găsesc în șirul menționat mai sus. Valorile determinate se afișează pe ecran în ordine strict crescătoare, separate prin câte un spațiu. Dacă nu există astfel de valori, se afișează pe ecran mesajul *Nu există*. Pentru determinarea valorilor cerute se utilizează un algoritm eficient din punctul de vedere al memoriei și al timpului de executare. (Examen de Bacalaureat - Iunie 2016)
12. Să se afișeze numerele dintr-un fișier text care au cifrele identice.
13. Din fișierul *Date.in* se citesc mai multe numere întregi, scrise pe unul sau mai multe rânduri. Să se copieze în fișierul *Date.out* numerele din fișierul de intrare care au cel puțin 3 cifre.
14. Fișierul *Numere.txt* conține pe prima linie 2 numere a și b , iar pe următoarea linie n numere naturale. Să se verifice câte dintre aceste numere se găsesc în intervalul $[a, b]$.
15. Fișierul *Bac.txt* conține numere naturale din intervalul $[1, 10^4]$: pe prima linie numărul n , pe a doua linie un șir de n numere ordonate strict descrescător, iar pe a treia linie două numere, x și y ($x \leq y$). Numerele de pe aceeași linie sunt separate prin câte un spațiu. Se cere să se afișeze pe ecran cel mai mare număr din șir care aparține intervalului $[x, y]$. Dacă nu există un astfel de număr, se afișează pe ecran mesajul *Nu există*. Pentru determinarea numărului cerut se utilizează un algoritm eficient din punctul de vedere al timpului de executare. (Examen de Bacalaureat - Simulare 2015)
16. Se citesc din fișierul *Date.txt* un șir de numere întregi aflate pe prima linie a fișierului. Să se afișeze câte dintre numerele citite sunt mai mici decât media aritmetică a tuturor numerelor din fișier.
17. Fișierul *Numere.in* conține numere întregi dispuse pe mai multe linii. Să se scrie un program care afișează în fișierul *Numere.out* numărul cel mai mic și cel mai mare de pe fiecare linie din fișierul inițial. Cele două numere se vor scrie pe aceeași linie, separate prin spațiu.
18. Fișierul *Cifre.txt* conține pe primul rând un șir de cifre. Să se formeze cu cifrele un număr (cifrele vor fi introduse în număr în ordinea în care apar în fișier).
19. Fișierul *Numere.in* conține pe primele două linii 2 șiruri de cifre separate prin spații, reprezentând două numere mari. Să se calculeze suma celor 2 numere și să se afișeze rezultatul în fișierul *Numere.out*.
20. Fie *Date.txt* un fișier text care conține mai multe linii de numere întregi, separate între ele printr-un spațiu. Să se determine câte dintre aceste numere sunt palindroame (numărul și oglinditul său sunt egale). Rezultatul se afișează pe ecran.
21. Se citesc n numere întregi din fișierul *Date.in*. Toate numerele sunt scrise în fișier pe primul rând. Să se scrie în fișierul *Date.out* numerele ordonate crescător astfel: pe primul rând se vor scrie elementele pozitive, iar pe al doilea, elementele negative.
22. Să se scrie într-un fișier text primele n numere prime. Se vor scrie câte k numere pe un rând (k dat), despărțite prin spațiu, excepție făcând ultima linie care poate avea mai puțin de k numere.
23. Fișierul text *Date.in* conține pe primele două rânduri două șiruri de numere de lungimi diferite. Să se interclaseze cele două șiruri, rezultatul fiind memorat în fișierul *Date.out*.
24. Fișierul *Secvențe.in* conține pe primul rând un șir de n numere întregi. Să se descompună șirul citit în cât mai puține subșiruri cu elemente ordonate crescător. Subșirurile astfel obținute vor fi afișate în fișierul *Secvențe.out* câte unul pe un rând.
25. Fie fișierul *Fracții.in* care conține pe fiecare linie câte două numere naturale, reprezentând numărătorul, respectiv numitorul unei fracții simple. Să se creeze fișierul *Fracții.out* care va conține pe fiecare rând numărătorii și numitorii fracțiilor ireductibile rezultate prin simplificarea fracțiilor din primul fișier.
26. Să se scrie în fișierul *Prime.txt* toate numerele prime de 3 cifre.
27. În fișierul *Numere.in* se află n numere naturale unul sub altul. Se cere să se verifice dacă numerele din fișierul *Numere.in* sunt consecutive (în ordinea dată). Se va scrie în fișierul *Numere.out* unul dintre mesajele *CONSECUTIVE*, respectiv *NECONSECUTIVE*.
28. În fișierul text *Date.in* pe prima linie se află trei numere naturale, primele două scrise în ordine crescătoare. Se cere să se scrie în fișierul *Date.out* toate numerele naturale cuprinse între primele două numere din fișierul *Date.in* și divizibile cu al treilea, câte 5 numere pe un rând (ultimul rând poate avea mai puține numere).
29. Fișierul *Date.in* conține un șir de cel mult un milion de numere naturale din intervalul $[0, 10^9]$, separate prin câte un spațiu. Șirul are cel puțin doi termeni pari și cel puțin doi termeni impari. Se cere să se afișeze pe ecran mesajul *DA* dacă șirul aflat în fișier are un subșir ordonat crescător, format din toți termenii pari ai săi și un subșir ordonat descrescător, format din toți termenii impari ai săi. Dacă nu există două astfel de subșiruri, programul afișează pe ecran mesajul *NU*. Pentru verificarea proprietății cerute utilizați un algoritm eficient din punctul de vedere al timpului de executare și al memoriei necesare. (Examen de Bacalaureat - Sesiunea Specială 2015)
30. Fișierul *Sortare.in* conține pe primul rând un număr natural n , iar pe următorul rând un șir de n numere întregi, separate prin spațiu. Să se inverseze ordinea elementelor cuprinse între elementul minim și elementul maxim din șirul dat. Șirul astfel ordonat se va scrie în fișierul *Sortare.out*.

31. În fișierul "Vector.in" se află pe prima linie valoarea lui n , iar pe linia a doua n componente ale unui vector, numere naturale separate între ele prin câte un spațiu. Se cere să se scrie în fișierul "Vector.out" toate componentele numere de 2 cifre, separate între ele printr-un spațiu.
32. Fișierul *Numere.in* memorează cel mult 1000000 de numere naturale cu cel mult nouă cifre. Numerele sunt ordonate strict crescător și separate prin câte un spațiu. Se consideră șirul 1, 4, 7 ... definit astfel: $f_1 = 1$, $f_2 = 4$ și $f_n = 2 \cdot f_{n-1} - f_{n-2}$, pentru $n > 2$. Se cere să se afișeze pe ecran numerele din fișier care sunt termeni ai șirului. Numerele sunt afișate în ordine strict crescătoare, separate prin câte un spațiu. Dacă nu există astfel de numere, se afișează pe ecran mesajul *Nu există*. Pentru determinarea numerelor cerute se utilizează un algoritm eficient din punctul de vedere al memoriei și al timpului de executare. Scrieți programul corespunzător algoritmului descris. (Examen de Bacalaureat - Model 2013)
33. În fișierul "Vector.in" se află pe prima linie valoarea lui n , iar pe linia a doua n componente ale unui vector (numere naturale cu maxim 9 cifre) separate între ele prin câte un spațiu. Se cere să se scrie în fișierul "Vector.out" toate componentele care sunt egale cu suma componentelor vecine (din stânga și din dreapta), separate între ele printr-un spațiu.
34. În fișierul "Zero.in" se află pe prima linie un număr natural n , iar pe următoarea linie n numere naturale. Să se determine numărul cifrelor nule cu care se termină numărul format din produsul celor n numere. Rezultatul va fi afișat în fișierul "Zero.out".
35. Fișierul *Bac.txt* conține un șir de cel puțin trei și cel mult 1000000 de numere naturale cu cel mult nouă cifre. Numerele din șir sunt separate prin câte un spațiu. Se cere să se afișeze pe ecran, separate printr-un spațiu, două numere distincte, anume cel mai mic număr par cu două cifre și cel mai mare număr par cu două cifre care *NU* fac parte din șir. Dacă nu există două astfel de valori, pe ecran se afișează mesajul *Nu există*. Pentru determinarea numerelor cerute se utilizează un algoritm eficient din punctul de vedere al timpului de executare. Scrieți programul corespunzător algoritmului descris. (Examen de Bacalaureat – Sesiunea specială 2013)
36. Se consideră fișierul text "Date.in" care conține mai multe linii cu numere întregi separate între ele printr-un spațiu. Pentru fiecare dintre aceste linii se va scrie în fișierul "Date.out" unul din mesajele *Crescător*, *Descrescător* sau *Nesortat*, dacă numerele din linia respectivă sunt ordonate crescător, descrescător sau nu sunt sortate.
37. Fișierul *Bac.txt* conține pe prima linie un număr natural, n ($n \in [2, 5000]$), și pe a doua linie un șir de $2n$ numere naturale din intervalul $[0, 5]$. Numerele aflate pe aceeași linie a fișierului sunt separate prin câte un spațiu. Se cere să se afișeze pe ecran valoarea obținută însumând toate produsele de forma $x * y$, unde x și y sunt numere de paritate diferită, x fiind printre primii n termeni ai șirului aflat în fișier, iar y printre ultimii n

- termeni ai acestui șir. Dacă nu există nici un astfel de produs, valoarea cerută este nulă. Pentru determinarea numărului cerut utilizați un algoritm eficient din punctul de vedere al timpului de executare și al memoriei necesare. (Examen de Bacalaureat - August 2015)
38. Fișierul *Bac.txt* conține pe prima linie un număr natural n ($1 \leq n \leq 10^6$), iar pe a doua linie cel mult 1000000 de numere naturale de forma 2^p ($0 \leq p \leq 9$), separate prin câte un spațiu. Se cere să se afișeze pe ecran numărul care ar apărea pe poziția n în șirul ordonat descrescător obținut din toate numerele aflate pe a doua linie a fișierului. Dacă șirul are mai puțin de n termeni, se afișează pe ecran mesajul *Nu există*. Pentru determinarea numărului cerut se utilizează un algoritm eficient din punctul de vedere al timpului de executare. (Examen de bacalaureat - Model 2015)
39. Se consideră un șir ai cărui termeni sunt numere naturale nenule, de o singură cifră. Numim *număr asociat* al acestui șir un număr natural format cu termenii șirului, în ordinea în care aceștia apar în șir. *Exemplu*: numărul asociat șirului 1, 2, 5, 3, 2 este 12532. Fișierul text *Bac.txt* conține un șir de cel puțin trei și cel mult 80 de termeni, numere naturale nenule, de o singură cifră, separate prin câte un spațiu. Se cere determinarea unui șir obținut prin eliminarea a doi termeni situați pe poziții consecutive în șirul aflat în fișier, astfel încât numărul asociat șirului obținut să fie maxim. Termenii șirului obținut se afișează pe ecran, separați prin câte un spațiu. Se utilizează un algoritm eficient din punctul de vedere al memoriei utilizate și al timpului de executare. (Examen de Bacalaureat - Simulare 2014)
40. Numim *inserare* a unui șir A într-un șir B introducerea, între două elemente ale șirului B , a tuturor elementelor lui A , pe poziții consecutive, în ordinea în care apar în A . Fișierul *Bac.in* conține numere naturale din intervalul $[1, 10^6]$: pe prima linie numerele m și n , iar pe fiecare dintre următoarele două linii câte un șir de m , respectiv de n numere întregi ordonate strict crescător. Numerele aflate pe aceeași linie a fișierului sunt separate prin câte un spațiu, iar numerotarea elementelor în șiruri începe de la 1. Se cere să se afișeze pe ecran poziția din al doilea șir începând de la care poate fi inserat primul șir, astfel încât șirul obținut să fie strict crescător. Dacă nu există o astfel de poziție, se afișează pe ecran mesajul *Imposibil*. Proiectați un algoritm eficient din punctul de vedere al spațiului de memorie utilizat și al timpului de executare. (Examen de Bacalaureat – Sesiunea Specială 2017)
41. Numim *secvență pară* într-un șir o succesiune de termeni ai șirului cu proprietatea că sunt numere pare și că se află pe poziții consecutive în șir; orice secvență are cel puțin doi termeni și este maximală în raport cu proprietatea precizată (dacă i se adaugă un alt termen, secvența își pierde această proprietate). Lungimea secvenței este egală cu numărul termenilor săi. Fișierul *Bac.txt* conține un șir de cel mult 10^6 numere naturale din

- intervalul $[0, 10^9]$. Numerele din șir sunt separate prin câte un spațiu. Se cere să se afișeze pe ecran numărul de secvențe pare de lungime maximă din șir. Proiectați un algoritm eficient din punctul de vedere al spațiului de memorie utilizat și al timpului de executare. (Examen de Bacalaureat - Iunie 2017)
42. Un șir de numere este o *progresie aritmetică de rație r* dacă oricare termen al său, cu excepția primului, se obține din cel care îl precede, prin adunarea la acesta a numărului r . Exemplu: șirul 12, 14, 16, 18, 20 este o progresie de rație 2. Fișierul *Bac.in* conține un șir de cel mult 10^6 numere naturale din intervalul $[0, 10^3]$, separate prin câte un spațiu. Se cere să se verifice dacă există un număr natural r , astfel încât toate numerele distincte din șir să poată fi rearanjate, pentru a forma o progresie aritmetică de rație r . Se afișează pe ecran numărul r , sau mesajul *NU*, dacă nu există un astfel de număr. Proiectați un algoritm eficient din punctul de vedere al timpului de executare.
43. Fișierul "*Permutare.in*" conține pe primul rând un șir de n numere naturale separate prin spațiu. Să se permute circular elementele șirului cu câte o poziție. Permutările astfel obținute vor fi afișate în fișierul "*Permutare.out*".
44. Un număr natural x , format din exact două cifre, este numit *sub-număr* al unui număr natural y dacă cifrele lui x apar, în aceeași ordine, pe ranguri consecutive, în numărul y . Exemplu: 21 este sub-număr al lui 12145, al lui 213, al lui 21, dar nu și al lui 123 sau al lui 231. Fișierul *Bac.txt* conține cel mult 1000000 de numere naturale din intervalul $[10, 10^9]$, separate prin câte un spațiu. Se cere să se afișeze pe ecran, separate prin câte un spațiu, *sub-numerele* care apar de cele mai multe ori în scrierea numerelor din fișier. Pentru determinarea *sub-numerelor* cerute se utilizează un algoritm eficient din punctul de vedere al timpului de executare. (Examen de Bacalaureat - Iunie 2014)
45. Din fișierul "*Secvență.in*" se citește un număr natural n și apoi se citesc n numere naturale. Să se afișeze în fișierul "*Secvență.out*", lungimea celei mai lungi secvențe din numerele citite care au proprietatea că începe și se termină cu aceeași valoare (nu mai conține acea valoare în afară de primul și ultimul element al secvenței).
46. Numim *secvență uniformă* a unui șir de numere naturale un subșir al acestuia, format din termeni cu aceeași valoare, aflați pe poziții consecutive în șirul dat. Lungimea secvenței este egală cu numărul de termeni ai acesteia. Fișierul *Bac.txt* conține un șir de cel puțin două și cel mult 1000000000 de numere naturale din intervalul $[0, 10^9]$. Numerele sunt separate prin câte un spațiu, iar în șir există cel puțin doi termeni egali pe poziții consecutive. Se cere să se determine o secvență uniformă de lungime maximă în șirul aflat în fișier și să se afișeze pe ecran lungimea acestei secvențe și, pe o linie nouă, separați prin câte un spațiu, termenii acesteia. Dacă sunt mai multe astfel de secvențe, se afișează doar termenii ultimei dintre acestea. Pentru determinarea numerelor cerute se utilizează un algoritm eficient din punctul de vedere al memoriei necesare și al timpului de executare. Scrieți programul corespunzător algoritmului descris. (Examen de Bacalaureat - Specială 2014)
47. Fiind date două numere a și b , îl numim pe a *sufix* al lui b dacă a este egal cu b sau dacă b se poate obține din a prin alipirea la stânga a unor noi cifre. Exemplu: 12 este sufix al lui 12, iar 15 este sufix al lui 31415. Fișierul *Bac.txt* conține pe prima linie un număr natural x , cu cel mult nouă cifre, iar pe a doua linie un șir de cel puțin două și cel mult 1000000 de numere naturale cu cel mult nouă cifre. Numerele din șir sunt separate prin câte un spațiu. Se cere să se afișeze pe ecran ultimul termen al șirului care are ca sufix numărul x . Dacă în șir nu există o astfel de valoare, pe ecran se afișează mesajul *Nu există*. Pentru determinarea numărului cerut se utilizează un algoritm eficient din punctul de vedere al memoriei și al timpului de executare. Scrieți programul corespunzător algoritmului descris. (Examen de Bacalaureat - August 2013)
48. Numim număr de tip *palindrom* un număr care are aceeași valoare dacă este citit de la stânga la dreapta sau de la dreapta la stânga. Numim număr de tip *dublu palindrom*, o valoare de tip palindrom cu număr par de cifre în care fiecare cifră de rang impar este egală cu cifra alăturată din dreapta ei. Exemplu: 11111111, 22777722 sunt numere de tip dublu palindrom; 12122121 este număr de tip palindrom, dar nu și dublu palindrom. Se cere scrierea în fișierul „*Bac.txt*”, fiecare pe câte o linie, a tuturor numerelor naturale de tip dublu palindrom, cu exact 8 cifre. Numerele de tip dublu palindrom sunt scrise în ordine strict crescătoare, iar pentru determinarea acestora se utilizează un algoritm eficient din punct de vedere al timpului de executare. Scrieți programul corespunzător algoritmului descris. (Examen de Bacalaureat - Model 2012)
49. Un număr natural cu cel puțin două cifre se numește *x -ordonat* dacă toate cifrele sale sunt în ordine crescătoare și valoarea absolută a diferenței dintre oricare două cifre aflate pe poziții consecutive este egală cu x . Exemple: numărul 2468 este 2-ordonat, numărul 147 este 3-ordonat; numerele 179 sau 131 nu sunt de tipul menționat. Se citește de la tastatură un număr natural x ($1 \leq x \leq 8$) și se cere scrierea în fișierul *Bac.txt* a tuturor numerelor naturale distincte *x -ordonate*. Fiecare număr este scris pe câte o linie a fișierului. Pentru determinarea numerelor cerute se utilizează un algoritm eficient din punctul de vedere al timpului de executare. Scrieți programul corespunzător algoritmului descris. (Examen de Bacalaureat - Iunie 2012)
50. Fișierul "*Matrice.txt*" conține pe primul rând două valori naturale m și n ($1 < n < 100, 1 < m < 100$), reprezentând numărul de linii respectiv de coloane ale unei matrice A , iar pe următoarele m linii, câte n valori

întregi cu maximum 4 cifre fiecare, separate prin câte un spațiu, reprezentând elementele matricei A . Se cere să se afișeze pe ecran un șir de $2(n + m) - 4$ numere ordonate crescător, șir format din elementele aflate pe chenarul exterior al matricei A . Chenarul exterior este format din prima linie, ultima linie, prima coloană și ultima coloană. Alegeți un algoritm de rezolvare eficient din punct de vedere al gestionării memoriei. (Variante Bacalaureat 2009).

51. Fișierul "*Matrice.txt*" conține pe prima linie un număr natural n ($2 < n < 20$), iar pe următoarele n linii, câte n valori întregi, despărțite prin spații, reprezentând elementele unui tablou bidimensional A . Numim *semidiagonală* a tabloului, mulțimea elementelor aflate pe o direcție paralelă cu diagonala secundară a acestuia. Scrieți un program care citește din fișierul "*Matrice.txt*" numărul n și elementele tabloului A și apoi citește de la tastatură două valori naturale L și C ($1 \leq L \leq n, 1 \leq C \leq n$). Programul va afișa pe ecran suma elementelor aflate pe semidiagonala ce conține elementul de pe linia L și coloana C . Se va considera că liniile și coloanele tabloului sunt numerotate începând cu 1.

52. Pentru o valoare n (număr natural, $1 < n < 100$) citită de la tastatură, scrieți un program care afișează în fișierul "*Matrice.txt*" un tablou bidimensional cu n linii și n coloane cu formatul de mai jos. Elementele de pe fiecare linie sunt separate prin spațiu.

1	0	0	0	...	0
2	1	0	0	...	0
3	2	1	0	...	0
...					
n	$n - 1$	$n - 2$	$n - 3$	...	1

(Variante Bacalaureat 2009)

53. În fișierul text "*Mat.txt*" se află $n * m$ ($1 < n < 11, 1 < m < 21$) valori numerice întregi de cel mult nouă cifre dispuse pe mai multe linii și separate prin spații. Valorile variabilelor n și m se citesc de la tastatură. Să se scrie un program care construiește în memorie și afișează pe ecran tabloul cu n linii și m coloane ce se formează cu elementele din fișier astfel încât ordinea tuturor elementelor din fișier să coincidă cu parcurgerea tabloului linie cu linie. (Variante Bacalaureat 2009)

54. Pentru codificarea unui text format numai din litere mici se folosesc două tablouri bidimensionale cu 5 linii și 5 coloane, după următorul exemplu:

a	b	c	d	e	1	2	3	4	5
f	g	h	i	j	6	7	8	9	10
k	l	m	n	p	11	12	13	14	15
o	r	s	t	u	16	17	18	19	20
v	w	x	y	z	21	22	23	24	25

Codificarea se face astfel:

- Caracterul din prima matrice de pe linia i și coloana j , a_{ij} se codifică prin elementul corespunzător din matricea cu numere naturale;
- După primul pas al codificării, linia i din tabloul de numere se permută spre dreapta cu k poziții, iar coloana j se permută în jos cu k poziții ($k < 5$).

Să se scrie un program, care pentru orice text dat, conținând doar litere mici ale alfabetului englez, să afișeze succesiunea de numere obținută prin codificare.

Rezultatul va fi trecut în fișierul text "*Codificare.out*", pe un rând, codurile fiecărei litere din textul citit inițial, fiind separate prin spațiu. Textul inițial va fi citit de la tastatură.

55. Se consideră o matrice pătratică A cu n linii și n coloane, cu p elemente nenule. Matricea este memorată sub forma a trei vectori E, L și C , care rețin valorile, liniile și coloanele elementelor nenule. Să se scrie un program care, citind vectorii E, L și C reface matricea A . Matricea astfel rezultată va fi afișată în fișierul "*Matrice.txt*". Vectorii inițiali vor fi citați din fișierul "*Vectori.txt*", astfel: pe primul rând în fișier se va găsi numărul n , iar pe următoarele trei rânduri, cei trei vectori. Elementele fiecărui vector vor fi separate prin spații.
56. Din fișierul "*Pătrate.in*" se citesc două numere naturale n și k ($n, k < 100$) și apoi se citesc 4 numere naturale x, y, l și c ($x, y, l, c < 100$) de pe fiecare din următoarele k linii ale fișierului. Semnificația acestora este: (x, y) sunt coordonatele stânga sus ale matricei (linia, respectiv coloana), l este lungimea, iar c este *culoarea* unui pătrat, reprezentată printr-un număr. Să se plaseze cele k pătrate într-o matrice pătratică cu n linii și n coloane și se afișeze în fișierul "*Pătrate.out*" matricea obținută. Datele de intrare asigură faptul că pătratele nu se suprapun și nu depășesc dimensiunile matricii. Elementele din matrice care nu fac parte din nici un pătrat vor avea valoarea 0.
57. Fișierul "*Date.in*" conține un șir de caractere. Scrieți un program care determină numărul de cifre din șir și îl afișează într-un fișier "*Date.out*".
58. Fișierul "*Date.in*" conține un șir de caractere. Scrieți un program care determină numărul de caractere speciale din șir și îl afișează într-un fișier "*Date.out*".
59. În fișierul "*Șir.in*" se află pe prima linie o propoziție cu maxim 255 caractere, iar pe linia a doua o altă propoziție. Se cere să se scrie în fișierul "*Șir.out*" propoziția cea mai lungă.
60. În fișierul text "*Vocale.in*" se află pe aceeași linie o propoziție. Se cere să se scrie în fișierul "*Vocale.out*" toate vocalele din fișierul "*Vocale.in*", una sub alta.
61. În fișierul text "*Consoane.in*" se află pe aceeași linie o propoziție. Se cere să se scrie în fișierul "*Consoane.out*" toate consoanele din fișierul "*Consoane.in*", pe un singur rând cu spații între ele.

62. Pe prima linie a fișierului text "*Bac.txt*" se găsește un cuvânt de exact 9 litere mici ale alfabetului englez. Pe a doua linie a fișierului se găsește o cifră nenulă k . Se cere să se scrie un program care afișează pe ecran, de k ori, cel de al k -lea caracter de pe prima linie a fișierului "*Bac.txt*".
63. Fișierul "*Fișier.txt*" conține pe prima linie un număr natural nenul n și pe fiecare dintre următoarele n linii câte un singur cuvânt format din cel mult 20 de litere mici ale alfabetului englez. Să se scrie un program care citește valoarea n și apoi determină câte dintre cele n cuvinte memorate în fișier conțin secvența *escu*. Programul va afișa pe ecran numărul de cuvinte determinat (0 în caz că nu există niciunul).
64. Scrieți un program care citește de la tastatură un șir cu maximum 255 de caractere format numai din litere mici și mari ale alfabetului englez și îl scrie în fișierul "*Cod.out*" codificat, conform următoarei reguli: dacă numărul de vocale (*a, e, i, o, u, A, E, I, O, U*) din șir este mai mare sau egal cu numărul de consoane din șir, fiecare vocală se va înlocui cu caracterul următor din alfabetul englez, altfel fiecare consoană se va înlocui cu caracterul precedent din alfabetul englez.
65. În fișierul "*Număr.in*" se află pe prima linie un singur număr natural. Se cere să se scrie în fișierul "*Număr.out*" cifrele numărului una sub alta.
66. În fișierul text "*Numere.in*", pe prima linie se află două numere scrise în ordine crescătoare. Se cere să se scrie în fișierul "*Numere.out*" toate numerele naturale cuprinse între cele două numere din fișierul "*Numere.in*", câte 10 numere pe un rând (ultimul rând poate avea mai puține numere).
67. Să se scrie un program care compară linie cu linie conținutul a două fișiere text, afișând numărul de ordine al liniilor identice.
68. Un fișier text "*Date.in*" conține pe prima linie un număr întreg n . Pe următoarele n linii se găsesc câte o literă c a alfabetului englez, urmată de un număr natural n , despărțite între ele printr-un spațiu. Să se construiască un fișier text "*Date.out*" care să conțină pe fiecare linie caracterul c repetat de n ori.
69. Prin asocierea fiecărei litere mici din alfabetul englez cu un număr egal cu poziția literei în alfabet, se poate codifica orice secvență de litere mici cu o secvență de numere. Scrieți un program care citește din fișierul text "*Cod.txt*" un număr n ($0 < n < 30000$) și, de pe rândul următor, o secvență de n litere mici. Să se scrie un program care va codifica secvența de caractere citită și va scrie pe ecran numerele asociate, separate prin spații.
70. Se dă fișierul "*Propoziții.in*" care conține pe fiecare rând cuvinte separate între ele prin unul sau mai multe spații. Să se creeze fișierul "*Propoziții.out*" care să conțină liniile primului fișier în ordinea crescătoare a numărului de cuvinte.
71. Se dă un fișier text "*Date.in*" care conține mai multe linii de text. Să se creeze un nou fișier "*Date.out*" care va conține pe fiecare linie numărul de vocale din linia corespunzătoare din primul fișier.
72. Se consideră fișierul text "*Cuvinte.in*" ce conține pe prima linie un număr natural nenul n ($n < 100$), iar pe următoarele n linii câte un cuvânt cu maximum 10 litere mici ale alfabetului englez. Să se afișeze pe ecran cuvintele din fișierul dat care *nu* sunt palindromuri. În cazul în care există numai cuvinte de tip palindrom se va afișa un mesaj corespunzător.
73. Realizați un program care *concatenează* două fișiere (adaugă conținutul celui de-al doilea fișier la sfârșitul primului).
74. Se dă un fișier "*Cuvinte.in*" care conține mai multe linii de text. Să se afișeze în fișierul "*Cuvinte.out*", pentru fiecare linie în parte, mulțimea caracterelor distincte.
75. Se dă fișierul "*Șir.txt*" care conține mai multe linii de text de lungime variabilă. Scrieți un program care afișează linia (liniile) de lungime maximă.
76. Să se copieze conținutul unui fișier text în alt fișier, înlocuind literele mici cu litere mari.
77. Se dă fișierul "*Șir.txt*" care conține mai multe linii cu cuvinte de lungime variabilă. Să se scrie un program care șterge fiecare apariție a unui șir s (citit de la tastatură) din fișierul text.
78. Fișierul "*Cuvinte.in*" conține una sau mai multe linii de cuvinte separate între ele prin unul sau mai multe spații. Să se copieze aceste cuvinte în fișierul "*Cuvinte.out*" câte 5 cuvinte pe o linie, despărțite între ele printr-un singur spațiu. Ultima linie din fișier poate avea mai puține cuvinte.
79. Fișierul "*Text.in*" conține un text, pe fiecare rând fiind scrisă o frază. Cuvintele din fraze pot fi separate prin virgulă sau spațiu. Să se afișeze în fișierul "*Text.out*" cuvintele distincte din text și frecvențele lor de apariție în cadrul textului (pe fiecare rând se va scrie un cuvânt împreună cu frecvența sa, separate printr-un spațiu).
80. Fișierul "*Cuvinte.txt*" conține pe câte un rând câte un cuvânt. Definim *randamentul* unui cuvânt ca fiind *numărul de caractere distincte din cuvânt / lungimea cuvântului*. De exemplu șirul "caractere" are randamentul $1/9=0.11$. Afișați în fișierul "*Rezultate.out*", randamentul cuvintelor citite – cu două zecimale. Pe fiecare rând din fișierul de ieșire vor fi trecute cuvântul și randamentul corespunzător lui.
81. Scrieți un program care citește din fișierul text "*Alfabetica.txt*" un cuvânt de cel mult 15 litere mici ale alfabetului englez și care scrie pe ecran mesajul *CORECT* dacă în cuvânt vocalele apar în ordine alfabetică (*a, e, i, o, u*) și niciuna dintre ele nu se repetă, iar în caz contrar scrie pe ecran mesajul *INCORECT*.
82. Scrieți un program care citește din fișierul "*Perechi.in*" n șiruri de caractere, oricare două distincte, formate din maximum 15 litere mici fiecare (fiecare șir de caractere va fi pe câte un rând în fișier). Să se creeze fișierul "*Perechi.out*" în care să se scrie toate perechile de șiruri dintre cele citite, perechi de forma (x, y) în care șirul x este subsecvență a șirului y sau

șirul y este subsecvență a șirului x . Fiecare pereche determinată se va scrie în fișierul "Perechi.out" pe câte un rând separate printr-o virgulă. Dacă nu există nici o astfel de pereche, în fișierul "Perechi.out" se va scrie textul **FĂRĂ SOLUȚIE**.

83. Fișierele text "Nume1.txt" și "Nume2.txt" conțin fiecare numele a n persoane, câte un nume pe fiecare linie, fiecare nume având cel mult 15 litere. Știind că în fiecare fișier numele sunt memorate în ordine alfabetică, scrieți un program care să citească din cele două fișiere și să afișeze pe ecran toate numele în ordine alfabetică, separate printr-un singur spațiu.
84. Scrieți un program care citește de la tastatură un șir s de cel mult 30 de litere și o literă c , care determină dublarea fiecărei apariții a literei c în s și scrie noul șir obținut în fișierul text "Dublare.txt".

3.7. Probleme olimpiadă

Problema 1 - Accesibil

Un număr natural de cel puțin două cifre se numește *accesibil* dacă este format din cifre consecutive în ordine strict crescătoare. (23 și 6789 sunt numere accesibile, în timp ce 7, 2334 și 654 nu sunt numere accesibile)

Cerințe:

Scrieți un program care să citească numerele k , n și un șir de n numere naturale și să afișeze:

- cele mai mari 3 numere accesibile, nu neapărat distincte, din șirul de n numere;
- câte dintre numerele din șirul dat care nu sunt accesibile, devin accesibile prin eliminarea exact a unei cifre;
- cel mai mic și cel mai mare număr accesibil format din k cifre;
- numărul numerelor accesibile pare de k cifre și numărul numerelor accesibile impare de k cifre.

(Olimpiada județeană, 2017)

Problema 2 - Tablou

Se consideră un tablou cu N linii și N coloane (numerotate de la 1 la N) care conține valoarea 1 în fiecare dintre cele $N \times N$ celule. Valorile din tablou pot fi modificate prin aplicarea a două operații codificate astfel:

- L nr**, prin care se schimbă simultan toate semnele numerelor din linia cu numărul **nr**.
- C nr**, prin care se schimbă simultan toate semnele numerelor din coloana cu numărul **nr**.

Cerințe

1) Dându-se o succesiune de **K** operații (**L nr** sau **C nr**) asupra liniilor/coloanelor tabloului inițial (în care toate celulele conțin valoarea 1) să se determine numărul valorilor pozitive din tablou la finalul executării celor **K** operații.

2) Să se determine numărul minim de operații **L nr** sau **C nr**, care, aplicate tabloului inițial, îl modifică astfel încât tabloul obținut să conțină exact **Z** valori negative.

Date de intrare

Fișierul **tablou.in** conține pe prima linie numărul **p=1** sau **p=2**, reprezentând numărul cerinței ce trebuie rezolvată.

- Dacă **p=1** atunci linia a doua a fișierului de intrare conține numerele **N** și **K**, separate printr-un spațiu, iar următoarele **K** linii conțin fiecare câte o literă mare (**L** sau **C**) și un număr **nr**, separate printr-un spațiu, reprezentând codificarea uneia dintre cele două operații (**L nr** sau **C nr**).
- Dacă **p=2** atunci linia a doua a fișierului de intrare conține numerele **N** și **Z**, separate printr-un spațiu.

Date de ieșire

- Dacă **p=1**, atunci fișierul de ieșire **tablou.out** conține pe prima linie un număr natural, reprezentând numărul valorilor pozitive din tabloul obținut la finalul executării celor **K** operații asupra tabloului inițial (răspunsul la cerința 1).
- Dacă **p=2**, atunci fișierul de ieșire **tablou.out** conține pe prima linie un număr natural reprezentând numărul minim de operații **L nr** sau **C nr**, care, aplicate tabloului inițial, îl modifică astfel încât tabloul obținut să conțină exact **Z** valori negative (răspunsul la cerința 2). Dacă prin aplicarea de operații **L nr** sau **C nr** tabloului inițial nu se poate obține un tablou cu **Z** valori negative, atunci, fișierul va conține pe prima linie valoarea 0 (zero).

Restricții și precizări:

- N**, **K**, **Z** și **nr** sunt numere naturale
- $3 \leq N \leq 20000$; $1 \leq K \leq 43000$; $1 \leq Z \leq N \cdot N$; $1 \leq nr \leq N$
- Prin schimbare de semn, valoarea -1 se transformă în 1 și valoarea 1 se transformă în -1
- Se acordă 10 puncte din oficiu și câte 45 puncte pentru rezolvarea corectă a fiecărei cerințe.

Exemplul 1

tablou.in tablou.out
1 10

4 4

L 1

L 3

C 1

L 1

Explicații

N=4. La finalul aplicării succesiunii de K=4 operații, tablou modificat are conținutul:

-1 1 1 1

-1 1 1 1

1 -1 -1 -1

-1 1 1 1

Astfel, tabloul conține 10 valori pozitive.

Exemplul al 2-lea

tablou.in tablou.out
2 3

3 5

Explicații

Sunt necesare minimum 3 operații, de exemplu:

L 3

L 1

C 1

Exemplul al 3-lea

tablou.in tablou.out
2 0

4 7

Explicații

Nu există nicio succesiune de operații pentru care să se obțină Z=7 valori negative.

(Olimpiada județeană, 2017)

Problema 3 - Dominant

Considerând un șir de valori binare, numim *secvență dominantă* un set de elemente aflate pe poziții consecutive în șir care are proprietatea că numărul valorilor egale cu 1 este strict mai mare decât numărul valorilor de 0. De exemplu, în șirul 1,0,0,0,1,1,0,1,1,1,0,0 o secvență dominantă este 0,1,1 și o alta, de lungime mai mare, este 0,1,1,0,1,1,1. Secvența dominantă maximală este secvența dominantă de lungime maximă. În șirul din exemplu secvența dominantă maximală este 1,0,0,0,1,1,0,1,1,1,0 (adică întreg șirul, fără ultimul zero).

Cerință

Dat un șir de valori binare, să se determine lungimea unei secvențe dominante maxime precum și numărul acestor secvențe.

Date de intrare

Fișierul **dominant.in** conține pe prima linie un număr natural V, iar pe linia a doua șirul de valori binare, fără spații.

Date de ieșire

Fișierul **dominant.out** va conține:

- varianta 1: dacă V=1, atunci pe prima linie a fișierului de ieșire va fi un singur număr natural reprezentând lungimea unei secvențe dominante maxime.
- varianta 2: dacă V=2, atunci pe prima linie a fișierului de ieșire va fi un singur număr natural reprezentând numărul secvențelor dominante maxime.

Restricții

- $V \in \{1,2\}$
- Lungimea șirului de valori binare este de cel mult 300 000.
- Pentru toate testele șirul binar va conține cel puțin o valoare de 1.
- Pentru 60% din punctaj $V = 1$.

Exemple

dominant.in	dominant.out	Explicații
1 100011011100	11	Secvența dominantă maximală este 10001101110 și are lungimea 11.
dominant.in	dominant.out	Explicații
2 100011011100	1	Secvența dominantă maximală este 10001101110 și are lungimea 11. Este o singură secvență dominantă maximală.
dominant.in	dominant.out	Explicații
1 0000110000111	9	Secvența dominantă maximală are lungime 9; aceasta este 110000111.
dominant.in	dominant.out	Explicații
2 10000111000	3	Secvența dominantă maximală are lungimea 5. Sunt trei secvențe dominante maxime: 00111, 01110 și 11100.

(Olimpiada județeană, 2015)

Problema 4 - Cool

Se consideră un șir A format din N elemente naturale nenule. Numim secvență de lungime K a șirului A orice succesiune de elemente consecutive din șir de forma $A_i, A_{i+1}, \dots, A_{i+K-1}$.

O secvență o numim **secvență cool** dacă elementele care o compun sunt distincte și pot fi rearanjate astfel încât să alcătuiască o secvență continuă de numere consecutive.

De exemplu, considerând șirul $A=(3,1,6,8,4,5,6,7,4,3,4)$, atunci secvența $(8,4,5,6,7)$ este o **secvență cool** deoarece conține elemente distincte ce pot fi rearanjate astfel încât să alcătuiască șirul de numere consecutive $4,5,6,7,8$, pe când secvențele $(4,3,4)$, $(6,7,4,3)$ **nu** sunt considerate **secvențe cool**.

Cerință

Fiind dat un șir de N numere naturale nenule se cer următoarele:

1. Pentru o valoare dată K să se verifice dacă secvența $A_1, A_2, \dots, A_K$ este **secvență cool**. Dacă secvența este cool, atunci se va afișa cea mai mare valoare ce aparține secvenței. Dacă secvența nu este cool, atunci se va afișa numărul elementelor distincte din secvența $A_1, A_2, \dots, A_K$, adică numărul elementelor care apar o singură dată.

2. Lungimea maximă a unei **secvențe cool** și numărul **secvențelor cool** de lungime maximă.

Date de intrare

Fișierul de intrare **cool.in** conține pe prima linie un număr natural p . Pentru toate testele de intrare, numărul p poate avea doar valoarea 1 sau valoarea 2. Pe linia a doua se găsesc, despărțite printr-un spațiu, două numere naturale N K . Pe următoarea linie se găsesc N numere întregi, separate prin câte un spațiu, ce reprezintă elementele șirului.

Date de ieșire

Dacă valoarea lui p este 1, atunci se va rezolva numai punctul 1 din cerință. În acest caz, fișierul de ieșire **cool.out** va conține pe prima linie un număr natural, număr ce reprezintă conform cerinței 1, maximul secvenței $A_1, A_2, \dots, A_K$, dacă secvența este **secvență cool**, sau numărul elementelor distincte din secvență, dacă aceasta **nu** este **secvență cool**.

Dacă valoarea lui p este 2, se va rezolva numai punctul 2 din cerință. În acest caz, fișierul de ieșire **cool.out** va avea două linii. Prima linie va conține un număr natural nenul ce reprezintă lungimea maximă a unei **secvențe cool**, iar următoarea linie un număr natural nenul ce reprezintă numărul de **secvențe cool** care au lungimea maximă.

Restricții și precizări

- $1 \leq N \leq 5000$
- $2 \leq K \leq 1000$
- $1 \leq A[i] \leq 1000, 1 \leq i \leq N$
- Pentru 30% dintre teste $N \leq 1000$
- Pentru rezolvarea primei cerințe se acordă 20% din punctaj, iar pentru cerința a doua se acordă 80% din punctaj.

Exemple

cool.in	cool.out	Explicație
1 7 4 6 4 5 7 8 3 5	7	Atenție! Pentru acest test se rezolvă doar cerința 1. Secvența 6 4 5 7 este cool. Valoarea maximă din secvență este 7
cool.in	cool.out	Explicație
1 7 6 6 4 5 7 5 4 3	2	Atenție! Pentru acest test se rezolvă doar cerința 1. Secvența 6 4 5 7 5 4 nu este secvență cool. Numărul valorilor distincte din secvență este 2. Valorile distincte sunt: 6,7
cool.in	cool.out	Explicație
2 11 4 7 4 5 6 8 4 5 7 4 3 2	5 2	Atenție! Pentru acest test se rezolvă doar cerința 2. Cele două secvențe cool de lungime maximă 5 sunt: 7 4 5 6 8 6 8 4 5 7

(Olimpiada județeană, 2014)

Problema 5 - Compar

Ana și Bogdan au inventat jocul "Compar". Ana scrie pe tablă o secvență formată din N numere naturale distincte cuprinse între 1 și N , apoi compară fiecare două numere învecinate din secvență scriind între ele semnul $<$ sau semnul $>$, după caz.

De exemplu, dacă secvența de pe tablă este 6 4 2 1 3 5, după compararea elementelor învecinate și inserarea semnelor în secvență, Ana obține:

6>4>2>1<3<5

După aceea Ana șterge cele N elemente ale secvenței și păstrează numai semnele, astfel:

>>><<

La final, Ana îi arată lui Bogdan șirul semnelor și îi cere să reconstituie secvența de numere naturale scrisă inițial pe tablă.

Cerință

Cunoscând șirul semnelor construit de Ana, scrieți un program care să îl ajute pe Bogdan să reconstituie secvența de numere naturale distincte scrisă inițial pe tablă.

Date de intrare

Fișierul de intrare compar.in conține pe prima linie o secvență de caractere din mulțimea $\{ '<', '>' \}$, reprezentând șirul semnelor obținut de Ana după compararea elementelor vecine din secvența inițială.

Date de ieșire

Fișierul de ieșire compar.out va conține pe prima linie numărul natural N , reprezentând lungimea secvenței inițiale. Pe a doua linie vor fi scrise N numere naturale distincte cuprinse între 1 și N , separate prin câte un spațiu, reprezentând elementele secvenței inițiale, reconstituită pe baza semnelor din fișierul de intrare.

Restricții

- $1 < N \leq 100000$
- Dacă există mai multe soluții, afișați oricare dintre acestea.
- Pentru determinarea corectă a lungimii secvenței se acordă 10% din punctajul pe test.

Exemplu

compar.in	compar.out	Explicație
>>><<	6	6>4>2>1<3<5
	6 4 2 1 3 5	Există și alte soluții posibile, aceasta este doar una dintre ele.

(Olimpiada județeană, 2013)

Problema 6 - Unific

Se consideră un șir $A=(A_1, A_2, \dots, A_N)$, format din N numere naturale nenule. Două numere se consideră vecine dacă se află pe poziții alăturate (A_i are ca vecini pe A_{i-1} și A_{i+1} , pentru orice $1 < i < N$, A_1 are ca vecin doar pe A_2 , iar A_N are ca vecin doar pe A_{N-1}).

Dacă două elemente vecine A_i, A_{i+1} ($1 \leq i < N$) au cel puțin o cifră comună, ele se pot unifica. Procedul de unificare constă în eliminarea din numerele A_i și A_{i+1} a tuturor cifrelor comune și adăugarea prin alipire a numărului obținut din A_{i+1} la numărul obținut din A_i , formându-se astfel un nou număr. Numărul A_i va fi înlocuit cu noul număr, iar numărul A_{i+1} va fi eliminat din șir. (De exemplu, numerele $A_i=23814$ și $A_{i+1}=40273$ au cifrele 2, 3, 4 comune, după unificare obținem $A_i=817$, iar A_{i+1} este eliminat; observați că dacă după eliminarea cifrelor comune, numerele încep cu zerouri ne semnificative, acestea vor fi eliminate, apoi se realizează alipirea). Dacă în urma eliminării cifrelor comune, unul dintre numere nu mai are cifre, atunci numărul rezultat va avea cifrele rămase în celălalt. Dacă în urma eliminării cifrelor comune atât A_i cât și A_{i+1} nu mai au cifre, atunci ambele numere vor fi eliminate din șir, fără a fi înlocuite cu o altă valoare.

Ordinea în care se fac unificările în șir este importantă: la fiecare pas se alege prima pereche de elemente vecine A_i, A_{i+1} care poate fi unificată, considerând șirul parcurs de la stânga la dreapta. (De exemplu, considerând $A_i=123$, $A_{i+1}=234$, $A_{i+2}=235$, se unifică A_i cu $A_{i+1} \Rightarrow A_i=14$, iar unificarea cu următorul număr nu mai este posibilă).

Cerință

Cunoscându-se șirul celor N numere naturale, să se determine:

- a) cifra care apare cel mai frecvent în scrierea tuturor celor N numere; dacă există mai multe cifre cu aceeași frecvență de apariție maximă, se va reține cea mai mică cifră.
- b) șirul obținut prin efectuarea unui număr maxim de unificări, după regulile descrise în enunț.

Date de intrare

Fișierul de intrare unific.in conține pe prima linie o valoare naturală N , iar pe următoarele N linii, în ordine, cele N numere naturale din șirul A , câte un număr pe o linie.

Date de ieșire

Fișierul de ieșire unific.out va conține pe prima linie un număr natural c reprezentând cifra care apare cel mai frecvent în scrierea celor N numere naturale. Pe cea de a doua linie un număr natural N_r reprezentând numărul de numere naturale rămase în șir după efectuarea unui număr maxim de unificări. Pe cea de a treia linie se vor scrie cele N_r numere naturale rămase, în ordinea din șir, separate prin câte un spațiu.

Dacă în urma procedurii de unificare, toate numerele vor fi eliminate, fișierul de ieșire va conține o singură linie, pe care se va scrie cifra care apare cel mai frecvent în scrierea celor N numere naturale.

Restricții

- $1 \leq N \leq 100\,000$
- Numerele din șirul inițial, precum și numerele obținute în urma unificărilor, nu vor depăși 10^{18}
- Pentru datele de test șirul obținut în urma unificărilor este nevid.
- Pentru 30% dintre teste $N \leq 1000$
- Pentru 70% dintre teste numere naturale din șir au cifrele nenule.
- Pentru determinarea corectă a primei cerințe se acordă 10% din punctajul pe test. Punctajul integral se acordă pe ambele cerințe rezolvate corect.

Exemplu

unific.in	unific.out	Explicații
10	3	Cifra care apare cel mai frecvent este 3 (de 6 ori).
6	2	Se unifică: 47 cu 67 $\Rightarrow$ 46. Șirul rămas: 6 46 40 123 231 1238 331 2035 50007
47	0 837	Se unifică: 6 cu 46 $\Rightarrow$ 4. Șirul rămas: 4 40 123 231 1238 331 2035 50007
67		Se unifică: 4 cu 40 $\Rightarrow$ 0. Șirul rămas: 0 123 231 1238 331 2035 50007
40		Se unifică: 123 cu 231, ambele numere rămân fără cifre, deci vor fi ambele eliminate.
123		Șirul rămas: 0 1238 331 2035 50007
231		Se unifică: 1238 cu 331 $\Rightarrow$ 28. Șirul rămas: 0 28 2035 50007
1238		Se unifică: 28 cu 2035 $\Rightarrow$ 835. Șirul rămas: 0 835 50007
331		Se unifică: 835 cu 50007 $\Rightarrow$ 837. Șirul rămas: 0 837
2035		
50007		

(Olimpiada județeană, 2013)

Problema 7 - Maxp

Considerăm un șir de numere $a_1, a_2, \dots, a_N$. O secvență nevidă în acest șir este de forma $a_i, a_{i+1}, \dots, a_j$, unde $i \leq j$. De exemplu, pentru $N=4$ și șirul 2 3 4 3, secvențele nevide sunt: 2, 2 3, 2 3 4, 2 3 4 3, 3, 3 4, 3 4 3, 4, 4 3, 3. Definim puterea unui element a_i ca fiind numărul de secvențe care-l conțin pe a_i și în care a_i este strict mai mare decât celelalte elemente ale fiecăreia dintre respectivele secvențe. Astfel în șirul 2 3 4 3 puterea elementului a_1 este 1 (fiind maxim doar în secvența formată din el însuși), a elementului a_2 este 2 (a_2 fiind maxim în secvențele 2 3 și 3), a elementului a_3 este 6 (fiind maxim în secvențele 2 3 4, 2 3 4 3, 3 4, 3 4 3, 4 și 4 3), iar a elementului a_4 este 1.

Cerințe

Scrieți un program care determină puterea cea mai mare a unui element din șirul dat, precum și numărul de elemente din șir care au cea mai mare putere.

Date de intrare

Fișierul **maxp.in** conține pe prima linie numărul natural N , iar pe a doua linie, în ordine, numerele naturale $a_1, a_2, \dots, a_N$ separate prin câte un spațiu.

Date de ieșire

Fișierul **maxp.out** va conține pe prima linie un număr natural ce reprezintă puterea cea mai mare a unui element din șirul dat și pe a doua linie va conține un număr natural ce reprezintă numărul de elemente din șir care au cea mai mare putere.

Restricții și precizări:

- $2 \leq N \leq 200\,000$
- Elementele șirului sunt numere naturale și au cel mult 6 cifre
- Se acordă 50% din punctaj pentru determinarea corectă a celei mai mari puteri a unui element din șir și 50% din punctaj pentru determinarea numărului de elemente din șir care au cea mai mare putere.

Exemplu

maxp.in	maxp.out	Explicații
7 9 3 4 5 1 2 2	12 1	Elementul 5 de pe poziția 4 este maxim în 12 secvențe: 3 4 5, 3 4 5 1, 3 4 5 1 2, 3 4 5 1 2 2, 4 5, 4 5 1, 4 5 1 2, 4 5 1 2 2, 5, 5 1, 5 1 2, 5 1 2 2, deci puterea lui este 12. Este singurul element care are această putere, celelalte elemente având puteri mai mici.
maxp.in	maxp.out	Explicații
6 1 0 7 7 2 6	3 2	Elementele din pozițiile 3 și 4 sunt maxime în 3 secvențe, deci puterea lor este 3. Celelalte elemente au puteri mai mici.

(Olimpiada județeană, 2013)

Problema 8 - Număr

Se dă un număr rațional strict pozitiv q , sub formă de fracție zecimală.

Cerință

Să se determine două numere naturale a și b astfel încât $q = \frac{a}{b}$ iar modulul diferenței dintre a și b să fie minim.

Date de intrare: fișierul **numar.in** conține

- pe prima linie două valori naturale n_1 și n_2 . n_1 reprezintă numărul de cifre care formează partea întreagă a lui q iar n_2 reprezintă numărul de cifre care formează partea fracționară a lui q .
- pe a doua linie, n_1 cifre care reprezintă partea întreagă a lui q . Între două cifre se află câte un caracter spațiu.
- pe a treia linie, n_2 cifre care reprezintă partea zecimală a lui q . Între două cifre se află câte un caracter spațiu.

Date de ieșire: fișierul **numar.out** va conține:

- pe prima linie un număr natural n_1 care reprezintă numărul de cifre din care este alcătuit numărul a ;
- pe a doua linie, cifrele numărului a , fără spațiu între ele.
- pe a treia linie un număr natural n_2 care reprezintă numărul de cifre din care este alcătuit numărul b ;
- pe a patra linie, cifrele numărului b , fără spațiu între ele.

Restricții și precizări:

- $1 \leq n_1, n_2 < 2000$
- $1 \leq n_1 + n_2 \leq 2000$
- Cifrele din care este alcătuit q sunt cele din sistemul zecimal.
- Pentru 20% dintre teste, $n_1 + n_2 \leq 9$; pentru alte 15% dintre teste, $10 \leq n_1 + n_2 \leq 16$

Exemple

numar.in	numar.out	Explicații
1 3	1	$q = 0.375 = \frac{3}{8}$
0	3	
3 7 5	1	Exista și alte perechi de valori naturale x, y
	8	astfel încât $\frac{x}{y} = 0.375$, dar pentru oricare altă
		pereche, $ x-y > 3-8 $ ($ \dots $ reprezintă modulul)
3 7	7	
0 1 2	1234567	$q = 12.34567 = \frac{1234567}{100000}$
3 4 5 6 7 0 0	6	
	100000	

(Olimpiada județeană, 2010)

Problema 9 - Cuvinte

Se consideră un șir de cuvinte separate două câte două printr-un spațiu. Fiecare cuvânt este caracterizat prin numărul de ordine care reprezintă poziția lui în șirul de cuvinte (primul cuvânt are numărul de ordine 1). Unui cuvânt i se pot aplica în mod repetat următoarele transformări: primul caracter al cuvântului (cel mai din stânga) se șterge de acolo și se adaugă după ultimul caracter din cuvânt. Astfel, dintr-un cuvânt s cu k caractere se pot obține alte $k-1$ cuvinte pe care le numim cuvinte obținute din transformarea cuvântului s . De exemplu, dintr-un cuvânt format din 4 caractere $c_1c_2c_3c_4$, cuvintele obținute prin transformarea lui sunt: $c_2c_3c_4c_1, c_3c_4c_1c_2, c_4c_1c_2c_3$.

Se caută în șirul de cuvinte prima pereche de cuvinte vecine (a,b) , în care al doilea cuvânt din pereche (cuvântul b) este identic cu un cuvânt obținut din transformarea lui a . Dacă există o astfel de pereche, se șterge cuvântul b din șir. Prin ștergerea cuvântului b din șir, acesta va avea mai puțin cu un cuvânt! Se repetă operația de căutare de mai sus până când în șirul rămas nu mai există o pereche (a,b) de cuvinte vecine, astfel încât b să fie obținut prin transformarea lui a .

Se știe că pe parcursul modificărilor, cuvintele nu-și schimbă numerele de ordine pe care le-au avut inițial.

Cerință

Scrieți un program care să citească șirul de cuvinte și să afișeze:

- numărul de ordine al primului cuvânt șters sau valoarea 0 în cazul în care nu se șterge niciun cuvânt
- numerele de ordine ale cuvintelor rămase după finalizarea operațiilor de modificare.

Date de intrare

Fișierul **cuvant.in** conține o singură linie pe care se află șirul de cuvinte separate două câte două printr-un spațiu.

Date de ieșire

Fișierul **cuvant.out** va conține:

- pe prima linie numărul de ordine al primului cuvânt șters sau valoarea 0 în cazul în care nu se șterge niciun cuvânt
- pe a doua linie numerele de ordine ale cuvintelor rămase în final în șirul de cuvinte, despărțite două câte două printr-un spațiu.

Restricții și precizări

- După ultimul cuvânt din șirul inițial există caracterul !
- Fiecare cuvânt are maxim 10 caractere, iar în șirul inițial nu există mai mult de 25 cuvinte.

- Șirul de cuvinte inițial este format din cel puțin un cuvânt. O pereche de cuvinte vecine (a,b), din șirul de cuvinte este caracterizată prin faptul că, după cuvântul a se afla imediat cuvântul b.
- Se acordă punctaje parțiale: cerința a) 40% din punctaj, cerința b) 60% din punctaj

Exemplu

cuvant.in	cuvant.out	Explicație
alfa faal alfa fala lafa afal calfa calfa!	2 1 3 4 7 8	Cuvintele obținute prin transformarea cuvântului alfa sunt: lfaa, faal, aalf . Prima pereche de cuvinte vecine care îndeplinesc cerințele sunt cuvintele cu numerele de ordine 1 și 2. Va fi șters cuvântul cu numărul de ordine 2. Șirul rezultat este format din următoarele 7 cuvinte: alfa alfa fala lafa afal calfa calfa . Prima pereche care îndeplinește cerințele din noul șir este perechea formată din cuvintele fala și lafa , având numerele de ordine 4 și 5, pentru că lista de cuvinte obținute prin transformarea cuvântului fala sunt: alaf, lafa, afal . Se va șterge cuvântul cu numărul de ordine 5. Șirul rezultat după ștergere este: alfa alfa fala afal calfa calfa . Prima pereche care îndeplinește cerințele problemei în noul șir este fala și afal . Se șterge afal cu numărul de ordine 6. Șirul rezultat după ștergere este: alfa alfa fala calfa calfa . În acest șir nu se mai găsește nicio pereche care să îndeplinească cerințele. Au rămas deci cuvintele: alfa, alfa, fala, calfa, calfa care au numerele de ordine 1, 3, 4, 7, 8.

(Olimpiada județeană, 2010)

Problema 10 – Secvențe

Mariei îi plac numerele prime și puterile numerelor prime. Pornind de la un număr prim p , ea construiește noi numere, fiecare număr construit fiind un produs de forma p^y ($y \in \mathbb{N}$, $y \neq 0$) sau $q \cdot p^m$, $m \in \mathbb{N}$ și q un număr prim, numindu-le numere p -prime. De exemplu, numerele 2, 3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 16, 17 sunt primele 13 numere 2-prime deoarece $2=2^1$, $3=3 \cdot 2^0$, $4=2^2$, $5=5 \cdot 2^0$, $6=3 \cdot 2^1$, $7=7 \cdot 2^0$, $8=2^3$, $10=5 \cdot 2^1$, $12=3 \cdot 2^2$, $13=13 \cdot 2^0$, $14=7 \cdot 2^1$, $16=2^4$, $17=17 \cdot 2^0$.

Într-o zi Maria a găsit o foaie de hârtie, pe care era scris un șir format din n numere naturale nenule. Cum pe lângă numerele p -prime ea este pasionată și de secvențe, și-a pus următoarea întrebare: câte secvențe sunt pe foaie cu următoarele proprietăți:

- conțin exact k numere p -prime;
- încep și se termină cu un număr p -prim.

În plus, Maria dorește să știe care este poziția de început și cea de final, pentru fiecare secvență descoperită, relative la șirul scris pe foaia de hârtie.

Cerință

Scrieți un program care să citească mai multe seturi de date, fiecare set fiind format din numerele n , p , k , cu semnificațiile din enunț, și șirul cu n elemente $a_1, a_2, a_3, \dots, a_n$, numerele Mariei. Programul va determina pentru fiecare set de date numărul secvențelor ce conțin exact k numere p -prime, precum și pozițiile de început și de final ale acestor secvențe în șirul din set.

Date de intrare

Pe prima linie a fișierului `secvente.in` se află numărul D reprezentând numărul de seturi de date din fișier. Seturile de date sunt scrise în fișier pe linii succesive. Pentru fiecare set de date, prima linie conține câte trei numere naturale: n (numărul de elemente de pe foaie), p și k (cu semnificația din enunț), separate prin câte un spațiu, iar fiecare dintre următoarele n linii conține câte un număr natural al șirului $a_1, a_2, a_3, \dots, a_n$, numerele din șirul Mariei.

Date de ieșire

Fișierul `secvente.out` va conține D soluții corespunzătoare celor D seturi de date. Pentru fiecare soluție prima linie va conține un număr x reprezentând numărul de secvențe ce îndeplinesc proprietățile cerute, iar fiecare dintre următoarele x linii vor conține câte 2 numere naturale, separate printr-un spațiu, reprezentând poziția de început, respectiv de final a fiecărei secvențe, linii ordonate crescător după poziția de început. Dacă în șir nu există o astfel de secvență, prima linie a setului va conține valoarea 0.

Restricții și precizări

- $1 \leq D \leq 15$;
- $1 \leq k \leq n \leq 15000$;
- $2 \leq p \leq 30000$; p este un număr natural prim
- $1 \leq a_1, a_2, a_3, \dots, a_n \leq 30000$; $a_1, a_2, a_3, \dots, a_n \in \mathbb{N}^*$ (pozițiile din șir sunt numerotate de la 1)

- numărul 1 nu este p-prim.
- o secvență dintr-un șir este formată din elemente aflate pe poziții consecutive în șirul dat.

Exemplu:

secvente.in	secvente.out	Explicații
2	2	Cum $D=2$, fișierul de intrare conține două seturi de date.
5 3 2	1 2	Primul set de date: $n=5$, $p=3$, $k=2$ și $a=(7, 27, 4, 45, 1)$. Șirul din acest set conține următoarele numere 3-prime: $a_1=7$ (număr prim), $a_2=27=3^3$ (putere a lui 3) și $a_4=45=5 \cdot 3^2$ (număr prim înmulțit cu o putere a lui 3). În șir sunt două secvențe cu câte 2 numere 3-prime: a_1, a_2 respectiv a_2, a_3, a_4 . Pe prima linie a fișierului de ieșire se va scrie valoarea 2, iar pe următoarele două linii, pozițiile de început și de final ale celor două secvențe determinate.
7	2 4	
27	0	Șirul a din al doilea set de date, $n=3$, $p=5$, $k=7$, $a=(3, 4, 5)$, nu conține nici o secvență cu proprietatea cerută. Astfel, în fișierul de ieșire, pe cea de-a patra linie, se va scrie valoarea 0.
4		
45		
1		
3 5 7		
3		
4		
5		

(Olimpiada județeană, 2010)

Problema 11 – Grad

Avem la dispoziție un text format din litere mici ale alfabetului englez și spații. În text cuvintele sunt separate prin unul sau mai multe spații. Fiecare literă din alfabet are asociat un număr ce reprezintă ordinea literei în alfabet. Astfel **a** are asociat numărul 1, **b** numărul 2, **c** numărul 3 și așa mai departe. Folosind această asociere definim *gradul* unui cuvând ca fiind suma numerelor asociate fiecărei litere. Spre exemplu cuvântul **bac** are gradul $2+1+3=6$.

Folosind cuvintele unui text putem forma grupe de cuvinte. Două cuvinte fac parte din aceeași grupă, dacă au același *grad*.

Cerință

Să se scrie un program care, pentru un text dat determină numărul de cuvinte și numărul de grupe.

Date de intrare

Fișierul de intrare **grad.in** are pe prima linie un număr natural n reprezentând numărul de caractere din text, iar pe linia a doua textul.

Date de ieșire

Fișierul de ieșire **grad.out** va conține pe prima linie numărul de cuvinte, iar pe linia a doua numărul de grupe.

Restricții și precizări

- $0 < n < 255$
- Textul conține cel puțin un cuvânt.
- Primul și ultimul caracter din text este literă.
- Pentru determinarea corectă a numărului de cuvinte se obține 30% din punctaj.
- Nu se obțin puncte dacă numărul de cuvinte este incorect.

Exemplu

grad.in	grad.out	Explicație
20	4	Textul conține 4 cuvinte și două grupe. Prima grupă este formată din cuvintele bac , aaac și bbb , care au fiecare gradul 6, iar a doua grupă conține doar cuvântul daca , cu gradul 9.
bac daca aaac bbb	2	

(Olimpiada județeană, 2009)

Problema 12– (235)

Definim o putere a lui 3 un număr de forma 3^k , (k număr natural strict pozitiv), o putere a lui 5 un număr de forma 5^k (k număr natural strict pozitiv) iar o putere a lui 2 un număr de forma 2^k (k număr natural strict pozitiv).

Se dă un șir de n numere naturale. Plecând de la acest șir, formăm un nou șir prin eliminarea tuturor numerelor care nu sunt puteri ale lui 3 și nici puteri ale lui 5. Ordinea relativă între numerele care nu sunt eliminate se păstrează.

Cerințe

Să se determine câte numere conține șirul nou format.

Să se determine deasemenea numărul de secvențe având lungimea egală cu o putere a lui 2 existente în șirul nou format în care numărul de puteri ale lui 3 este egal cu numărul de puteri ale lui 5. O secvență este formată din elemente aflate pe poziții consecutive în acest șir nou format iar lungimea unei secvențe este egală cu numărul de elemente pe care aceasta îl conține.

Date de intrare

Pe prima linie în fișierul **235.in** se afla un număr natural **n**. Pe fiecare dintre următoarele **n** linii câte un număr natural mai mare decât **1** reprezentând numerele șirului inițial.

Date de ieșire

Pe prima linie a fișierului **235.out** se va afla o valoare naturală **m** care va reprezenta numărul de elemente rămase în șir după eliminare. Pe a doua linie se va afla o valoare naturală **s** reprezentând numărul de secvențe din șirul nou format care au proprietățile cerute.

Restricții și precizări

- $2 \leq n \leq 500000$
- Numerele din șirul inițial sunt numere naturale din intervalul $[2, 2000000000]$.
- Se garantează că $m \leq 40000$ pentru fiecare set de date de intrare.
- Pentru determinarea corectă a valorii numărului **m** se acordă 30% din punctaj iar pentru determinarea corectă a ambelor valori (**m** și **s**) se acordă 100% din punctaj.

Exemplu:

235.in	235.out	Explicații
8	6	Șirul rămas după eliminarea numerelor care nu sunt puteri ale lui 5 sau ale lui 3 are 6 elemente:
625	4	625, 125, 5, 9, 81, 125.
125		În acest șir sunt:
5		- două secvențe formate din două
9		valori care conțin un număr egal de
15		puteri ale lui 3 și ale lui 5: 5, 9 și
81		81, 125;
100		- două secvențe de patru numere care
125		conțin un număr egal de puteri ale lui
		3 și ale lui 5: 125, 5, 9, 81 și 5, 9, 81,
		125

(Olimpiada județeană, 2009)

Problema 13 – Vecini buni

Se consideră matricea **A** ale cărei elemente pot avea doar valorile **0** sau **1** și în care numerotarea liniilor și numerotarea coloanelor începe de la **1**. Pentru un element oarecare al matricei, definim noțiunea de vecin ca fiind acele elemente din matrice aflate în imediata sa apropiere, pe una dintre direcțiile orizontală, verticală sau pe cele două diagonale (vezi figura de mai jos în care s-au marcat cu **x** vecinii elementului marcat cu **o**). Un vecin bun al elementului **A[i][j]** este un vecin care are aceeași valoare cu **A[i][j]**.

```
x x x
x o x
x x x
```

Cerință

Dându-se matricea **A**, să se determine numărul maxim de vecini buni pe care îi are unul dintre elementele matricei precum și numărul de elemente care au acest număr maxim de vecini buni.

Date de intrare

Fișierul de intrare **vecini.in** conține pe prima linie trei valori naturale **m**, **n** și **k** reprezentând numărul de linii, numărul de coloane și numărul de valori egale cu **1** din matricea **A**. Pe fiecare dintre următoarele **k** linii ale matricei se află două valori **i** și **j** cu semnificația **A[i][j]** este egal cu **1**. Aceste valori sunt date în ordinea parcurgerii matricei pe linii de la linia **1** la linia **m**.

Date de ieșire

Fișierul de ieșire **vecini.out** va conține pe prima linie două numere naturale **x** și **y** separate printr-un singur spațiu: **x** va reprezenta numărul maxim de vecini buni pe care îi are unul dintre elementele matricei date iar **y** va reprezenta numărul de elemente din matricea dată care au acest număr maxim de vecini buni.

Restricții și precizări

- $2 \leq m, n \leq 1000$
- $0 \leq k \leq n * m$
- pentru 30% dintre teste, $2 \leq m, n \leq 200$

Exemplu:

vecini.in	vecini.out	Explicații
3 4 7	3 6	matricea A este următoarea:
1 3		<u>0 0 1 0</u>
2 1		<u>1 0 0 1</u>
2 4		<u>1 1 1 1</u>
3 1		Numărul maxim de vecini buni pe care îl
3 4		întâlnim pentru elementele matricei este 3. Sunt 6
3 3		elemente care au acest număr maxim de vecini
3 2		buni, cele situate în pozițiile: <u>1,2</u> ; <u>2,2</u> ; <u>2,3</u> ; <u>2,4</u> ;
		<u>3,2</u> ; <u>3,3</u> (cele care sunt subliniate în reprezentarea
		de mai sus a matricei).

(Olimpiada județeană, 2009)

Problema 14– Expresie

Costel are de rezolvat o temă grea la matematică: având la dispoziție N numere naturale nenule trebuie să așeze între acestea 2 operații de înmulțire și $N-3$ operații de adunare, astfel încât rezultatul calculelor să fie cel mai mare posibil. Nu este permisă modificarea ordinii numerelor date.

De exemplu, dacă $N=5$ și numerele sunt 4, 7, 1, 5, 3, operațiile pot fi așezate $4 + 7 * 1 + 5 * 3, 4 * 7 * 1 + 5 + 3$ e.t.c

Cerință

Scrieți un program care să așeze două operații de înmulțire și $N-3$ operații de adunare între cele N valori date astfel încât valoarea expresiei obținute să fie maximă.

Date de intrare

Fișierul de intrare **expresie.in** are următoarea structură:

Pe prima linie se află un număr natural N , reprezentând numărul elementelor date.

Pe următoarele linii se află cele N numere naturale date, fiecare pe câte o linie.

Date de ieșire

Fișierul de ieșire **expresie.out** va conține, pe prima linie, valoarea maximă obținută prin evaluarea expresiei.

Restricții și precizări

- $4 \leq N \leq 1000$
- Numerele date sunt numere naturale între 1 și 10000

Exemplu

expresie.in	expresie.out	Explicație
5	44	Valoarea maximă se obține prin așezarea
4		operațiilor sub forma:
7		$4 * 7 + 1 + 5 * 3$
1		
5		
3		

(Olimpiada județeană, 2009)

Problema 15 – Placare

O suprafață dreptunghiulară de înălțime N și lățime M unități trebuie acoperită perfect (placată) prin utilizarea unor plăci de formă dreptunghiulară de dimensiune $1 \times P$ sau $P \times 1$, unde P este un număr natural nenul. Suprafața dată poate fi privită ca un caroiaj cu $N \times M$ pătrățele egale cu unitatea.

O placare corectă a suprafeței inițiale se memorează într-un fișier text folosind următoarele convenții de codificare:

- pe prima linie se precizează dimensiunile N și M ale suprafeței;
- o placă dreptunghiulară de lățime P este codificată prin numărul natural P , iar o placă de înălțime P se codifică prin numărul întreg $-P$;
- convenim ca placa având ambele dimensiuni egale cu unitatea să se codifice cu valoarea 1;
- pe fiecare din cele N linii ale codificării se află câte un șir de valori întregi reprezentând, în ordine de la stânga la dreapta, codurile plăcilor care se găsesc amplasate începând de la respectiva linie;
- codul P strict mai mare ca 1 al unei plăci orizontale apare o singură dată pe linia corespunzătoare pe care se află placa, iar codul $-P$ al unei plăci verticale va apare o singură dată și anume pe prima linie de la care placa respectivă este amplasată în jos pe o anumită coloană a suprafeței;
- Dacă pe o anumită linie a suprafeței nu există astfel de coduri de plăci, atunci pe respectiva linie din fișier este o singură valoare de 0.

Folosind codificarea unei plăcări a suprafeței inițiale, se poate determina imaginea acestei plăcări sub forma unui tablou bidimensional A , cu N linii și M coloane, unde A_{ij} = valoarea absolută a codului plăcii care se suprapune peste pătrățelul de pe linia i și coloana j .

Cerință

Cunoscând codificarea unei plăci corecte a suprafeței date să se obțină imaginea acestei plăci (matricea de valori corespunzătoare codificării suprafeței).

Date de intrare

Fișierul de intrare `placare.in` are următoarea structură:

- pe prima linie valorile naturale N și M , separate printr-un spațiu, unde N este înălțimea suprafeței, M este lățimea suprafeței.
- pe fiecare din următoarele N linii se află un șir de valori întregi, separate prin câte un spațiu, reprezentând codificarea respectivei linii a plăcii.

Date de ieșire

În fișierul de ieșire `placare.out` se va tipări tabloul bidimensional ce reprezintă imaginea plăcii, compus din N linii, pe fiecare dintre ele aflându-se M valori naturale separate prin câte un spațiu, cu semnificația din enunț.

Restricții și precizări

- $1 \leq N, M \leq 100$ pentru 80% din teste; $100 < N, M \leq 300$ pentru 20% din teste
- dimensiunea P sau $-P$ a unei plăci este aleasă astfel încât acoperirea obținută să nu depășească înălțimea N sau lățimea M a suprafeței.
- datele din fișierul de intrare sunt corecte în sensul că reprezintă codificarea unei acoperiri a zonei dreptunghiulare de dimensiuni N și M .

Exemplu

placare.in	placare.out	Explicații
4 4	4 1 1 1	Valoarea -4 codifică o placă de înălțime 4 și lățime 1 plasată începând din pătratul de coordonate (1,1) și pînă în pătratul de coordonate (4,1)
-4 1 1 1	4 1 2 2	
1 2	4 2 2 1	
2 1	4 3 3 3	Valoarea 3 de pe ultima linie a codificării desemnează o placă de lățime 3 și înălțime 1, plasată orizontal, începând din pătrățelul de coordonate (4,2).
3		
3 2	3 2	
-3 -2	3 2	
0	3 1	
1		

Problema 16- Admitere

Să ne imaginăm faptul că la un anumit liceu există doar două clase per generație: una de Real și una de Uman. În prezent au loc înscrierile pentru clasa a IX-a. Cele două clase au fiecare câte M locuri disponibile, atât la Real, cât și la Uman. Dacă lista de elevi înscriși la o anumită clasă conține mai mult de M elevi, vor fi admiși acei M elevi care au notele cele mai mari. Ambele clase au deja M elevi înscriși, iar pentru fiecare se știe nota cu care a fost înscris la clasa respectivă.

Mai există însă N elevi, singurii încă neînscriși, care sunt privilegiați în acest proces (fiindcă au terminat gimnaziul la acest liceu). Privilegiul lor constă în următorul fapt: ei se pot înscrie acum, după ce înscrierile publice au fost încheiate, și se cunosc notele de înscriere la ambele clase. Fiecare din cei N elevi are câte două note: nota cu care ar fi înscris la Real și nota cu care ar fi înscris la Uman (acestea pot fi diferite, deoarece examenele de admitere de la cele două clase diferă). Fiecare din cei N elevi va alege să se înscrie în maxim o clasă. Ei își vor coordona alegerile astfel încât să **maximizeze** numărul de elevi admiși. Deoarece calculele devin destul de complicate, aceștia s-ar putea folosi de ajutorul vostru. Ei doresc răspunsul la următoarele două întrebări:

Cerințe:

(1) Care este numărul maxim de elevi **privilegiați** care pot fi admiși dacă se pune restricția suplimentară ca toți elevii privilegiați admiși să fie admiși la aceeași clasă?

(2) Care este numărul maxim de elevi **privilegiați** care pot fi admiși dacă aceștia se pot înscrie la clase diferite?

Date de intrare

Fișierul de intrare `admitere.in` conține pe primul rând o valoare egală cu 1 sau 2, reprezentând cerința ce urmează a fi rezolvată. Următoarea linie conține cele două numere N și M . Pe al treilea rând se află M numere, separate prin câte un spațiu, reprezentând notele cu care au fost înscriși elevii care formează momentan clasa de Real. Pe al patrulea rând se află M numere, separate prin câte un spațiu, reprezentând notele cu care au fost înscriși elevii care formează momentan clasa de Uman. Următoarele N linii vor conține câte o pereche de numere $R[i]$, $U[i]$, separate prin câte un spațiu, reprezentând nota cu care al i -lea elev privilegiat s-ar înscrie la clasa de Real, respectiv la clasa de Uman.

Date de ieșire

Fișierul de ieșire `admitere.out` va conține pe prima linie valoarea **MAX**: numărul maxim de elevi privilegiați admiși. A doua linie va conține un șir de N caractere din mulțimea {'R', 'U', 'X'}, care va descrie scenariul optim. Dacă al i -lea elev va fi înscris la Real, al i -lea caracter va fi egal cu 'R'. Dacă al i -lea elev va fi înscris la Uman, al i -lea caracter va fi egal cu 'U'. Dacă acesta nu va fi înscris nicăieri, al i -lea caracter va fi egal cu 'X'.

Deoarece elevii nu vor să depună efort inutil, un elev privilegiat care nu va fi admis în scenariul optim nu se va înscrie la nicio clasă. Cu alte cuvinte, pentru ca scenariul descris să fie considerat corect este necesar ca **exact MAX** caractere din șir să fie diferite de 'X'.

Restricții și precizări

- $1 \leq N, M \leq 2000$
- Teste în valoare totală de **25** de puncte vor solicita rezolvarea cerinței (1), iar restul de **65** de puncte vor solicita rezolvarea cerinței (2). Din oficiu sunt acordate **10** puncte.
- Pentru cerința (2), teste în valoare totală de **45** de puncte vor avea $1 \leq N, M \leq 150$
- Toate cele $N + M$ note pentru clasa de Real sunt distincte două câte două. Același lucru este valabil și în cazul notelor pentru clasa de Uman.
- Toate notele sunt numere naturale din intervalul **[1, 4000]**.
- Notele elevilor deja înscriși de la clasa de Real, respectiv Uman vor fi date în ordine crescătoare.
- În cazul în care există mai multe soluții corecte, este acceptată oricare dintre acestea.

Exemplu:

admitere.in	admitere.out	Explicație
1 2 3 2 4 6 6 7 8 3 5 12 14	1 XR	Nu este posibil ca ambii elevi să fie admiși la aceeași clasă. Există mai multe soluții în care un singur elev este admis: XR, XU, RX. Oricare din acestea este corectă.
2 2 3 2 4 6 6 7 8 3 5 12 14	2 RU	Deoarece acum rezolvăm cerința (2), ne este permis să înscriem elevii la clase diferite. Există o soluție în care ambii elevi sunt admiși, iar aceasta este unică: cea în care elevul 1 este înscris la Real (el nu putea fi admis la Uman indiferent de decizia celui de-al doilea elev), iar cel de-al doilea elev este înscris la Uman.

(Olimpiada județeană, 2017)

Problema 17– Roboți

Ștefan a împlinit 15 ani. Fiind un pasionat membru al Clubului de Robotică, familia i-a dăruit de ziua lui foarte mulți roboți, fiecare dotat cu o armă de o anumită putere. El a așezat toți roboții în jurul său, pe circumferința unui cerc imaginar, în sensul acelor de ceasornic. Aceste dispozitive inteligente pot comunica între ele, unindu-și puterile armelor.

Cerințe

Cunoscând numărul de roboți, precum și puterea fiecăruia, să se scrie un program care determină:

1. Dimensiunea celei mai lungi secvențe de roboți pentru care puterile armelor lor formează un șir strict crescător.
2. O aranjare a roboților pe cerc, astfel încât suma produselor de câte două puteri vecine să fie maximă. Dacă există mai multe modalități de aranjare astfel încât să se obțină aceeași sumă maximă, se va determina cea minimă din punct de vedere lexicografic.

Date de intrare

Pe prima linie a fișierului de intrare **roboți.in** se găsește un număr natural v a cărui valoare poate fi doar **1** sau **2**.

Pe a doua linie a fișierului de intrare se găsește un singur număr natural n reprezentând numărul de roboți.

Pe a treia linie a fișierului de intrare se găsesc n numere naturale $p_1, p_2, \dots, p_n$, separate prin câte un spațiu, p_i reprezentând puterea armei robotului i .

Date de ieșire

Dacă valoarea lui v este **1**, atunci fișierul de ieșire **roboți.out** va conține pe prima linie un singur număr natural reprezentând dimensiunea celei mai lungi secvențe de roboți pentru care puterile armelor lor formează un șir strict crescător.

Dacă valoarea lui v este **2**, atunci fișierul de ieșire va conține pe prima linie n numere naturale separate prin câte un spațiu, reprezentând puterile celor n roboți așezați pe cerc astfel încât suma produselor de câte două puteri vecine să fie maximă, iar așezarea să fie minimă din punct de vedere lexicografic.

Restricții și precizări

- $2 \leq n \leq 100000$
- Pentru cerința 1, secvența de lungime maximă se alege pe cerc în sensul acelor de ceasornic.
- Dacă avem două șiruri de numere $a_1, a_2, \dots, a_n$ și $b_1, b_2, \dots, b_n$ și există $1 \leq k \leq n$, cea mai mică poziție, pentru care are loc $a_1 = b_1, a_2 = b_2, \dots, a_{k-1} = b_{k-1}$ și $a_k < b_k$, atunci spunem că șirul a este mai mic lexicografic decât șirul b .
- Pentru rezolvarea corectă a cerinței **1** se acordă **30** puncte, pentru rezolvarea corectă a cerinței **2** se acordă **60** de puncte, iar din oficiu se acordă **10** puncte.

- Pentru cerința 2, dacă soluția afișată nu este minimă lexicografic, dar produce suma maximă corectă se acordă 50 % din punctajul testului respectiv.
- Pentru cerința 2, teste în valoare totală de 36 puncte vor avea $n \leq 1000$.
- $1 \leq p_1, p_2, \dots, p_n \leq 1000$.

Exemple

roboti.in	roboti.out	Explicații
1 7 4 7 2 6 5 1 3	4	$v = 1$, deci se va rezolva DOAR prima cerință. <u>4 7 2 6 5 1 3</u> Cea mai lungă secvență strict crescătoare este 1 3 4 7 și are lungimea 4.
2 5 3 9 1 12 5	1 3 9 12 5	$v = 2$, deci se va rezolva DOAR a doua cerință. $1*3+3*9+9*12+12*5+5*1=203$, și este suma maximă care se poate obține. Această aranjare nu este singura pentru care se obține suma maximă, dar este cea mai mică lexicografic.
2 4 1 2 1 1	1 1 1 2	$v = 2$, deci se va rezolva DOAR a doua cerință. $1*1+1*1+1*2+2*1=6$, și este suma maximă care se poate obține. Această aranjare nu este singura pentru care se obține suma maximă, dar este cea mai mică lexicografic.

(Olimpiada județeană, 2017)

Problema 18 – Arc

Irinuca a descoperit un nou joc pe calculator. Pe ecran sunt plasate pe o linie n bile colorate. Culoarele bilelor sunt codificate cu numere naturale. Un subșir de bile alăturate având toate aceleași culoare se numește *secvență*. O secvență va conține numărul maxim de bile alăturate având aceleași culoare. Lungimea unei secvențe este egală cu numărul de bile din care este compusă.

Irinuca are la dispoziție un arc special. Trăgând cu arcul asupra unei bile, dacă aceasta face parte dintr-o secvență de lungime cel puțin egală cu 3, întreaga secvență va fi eliminată, iar bilele din dreapta secvenței se vor deplasa spre stânga pentru a umple “golul” lăsat de bilele eliminate. Dacă imediat în stânga și în dreapta secvenței eliminate se găseau două secvențe având aceleași culoare și dacă secvența obținută din unirea acestora după eliminare are o lungime cel puțin egală cu 3, atunci va fi și ea eliminată la rândul ei. Procesul continuă până când secvențele din stânga și dreapta unei secvențe tocmai eliminate au culori

diferite sau până când lungimea secvenței obținute prin alăturare este mai mică decât 3 sau până când în stânga ori în dreapta unei secvențe eliminate nu se mai găsesc bile sau până sunt eliminate toate bilele de pe ecran.

Scopul jocului este de a elimina cât mai multe bile de pe ecran. Cum Irinuca încă nu se pricepe prea bine la acest joc și-a stabilit o strategie. Va trage cu arcul întotdeauna asupra unei bile ce face parte din secvența de lungime maximă de pe ecran. Dacă sunt mai multe astfel de secvențe, ea va alege cea mai din stânga secvență de lungime maximă. Dacă toate secvențele de pe ecran au lungimi mai mici decât 3, Irinuca nu va mai putea elimina nici una din ele și jocul se încheie.

De exemplu, dacă șirul inițial de bile este

5 1 3 3 2 2 2 2 3 1 1 5 6 4 4 4 4 7

Irinuca va acționa asupra unei bile de culoare 2. Prin eliminare se obține șirul de bile

5 1 3 3 3 1 1 5 6 4 4 4 4 7

din care se elimină și secvența de bile de culoare 3 obținându-se șirul de bile

5 1 1 1 5 6 4 4 4 4 7

din care se elimină și secvența de culoare 1.

5 5 6 4 4 4 4 7

Cum secvența de bile de culoare 5 nu este suficient de lungă, aceasta nu se mai elimină. Acum Irinuca trage asupra unei bile de culoare 4 și obține

5 5 6 7

dar cum în stânga și în dreapta secvenței eliminate sunt secvențe de culori diferite, nu se va mai elimina nici o secvență. Jocul se încheie deoarece nu mai există nici o secvență de lungime cel puțin 3 asupra căreia să se poată trage.

Cerință

Cunoscând numărul de bile și culorile fiecărei bile de pe ecran se cere să se determine:

1. numărul de secvențe de bile care se aflau inițial pe ecran;
2. numărul de bile care rămân neeliminate de pe ecran și culorile bilelor rămase în ordine pe ecran la finalul jocului.

Date de intrare

Fișierul de intrare **arc.in** conține pe prima linie un număr natural V . Pentru toate testele de intrare, numărul V poate avea doar valoarea 1 sau 2.

A doua linie conține un număr natural n reprezentând numărul de bile, iar a treia linie conține n numere naturale $c_1, c_2, \dots, c_n$ separate prin câte un spațiu, reprezentând culorile celor n bile de pe ecran.

Date de ieșire

Dacă valoarea lui **V** este **1**, se va rezolva numai **punctul 1** din cerință.

În acest caz, în fișierul de ieșire **arc.out** se va scrie un singur număr natural **n₁**, reprezentând numărul de secvențe de bile aflate inițial pe ecran.

Dacă valoarea lui **V** este **2**, se va rezolva numai **punctul 2** din cerință.

În acest caz, în fișierul de ieșire **arc.out** se va scrie pe prima linie un singur număr natural **n₂**, reprezentând numărul de bile care rămân neeliminate de pe ecran la finalul jocului, iar pe următoarele **n₂** linii se va scrie câte un număr natural reprezentând în ordine culorile bilelor rămase neeliminate la finalul jocului.

Dacă la finalul jocului nu mai rămâne nici o bilă neeliminată, fișierul de ieșire va conține pe prima sa linie valoarea **0**.

Restricții și precizări

- $1 \leq n \leq 10\,000$
- $1 \leq c_1, c_2, \dots, c_n \leq 100\,000$
- Pentru rezolvarea corectă a primei cerințe se acordă **20** de puncte, iar pentru cerința a doua se acordă **80** de puncte.

Exemple

arc.in	arc.out	Explicație
1 18 5 1 3 3 2 2 2 2 3 1 1 5 6 4 4 4 4 7	10	V = 1 Atenție! Pentru acest test se rezolvă doar cerința 1. Secvențele sunt (5), (1), (3, 3), (2,2,2,2), (3), (1,1), (5), (6), (4,4,4,4), (7)
arc.in	arc.out	Explicație
2 18 5 1 3 3 2 2 2 2 3 1 1 5 6 4 4 4 4 7	4 5 5 6 7	V = 2 Atenție! Pentru acest test se rezolvă doar cerința 2.

arc.in	arc.out	Explicație
2 15 1 2 2 2 2 1 1 3 3 3 4 4 4 4 3	0	V = 2 Atenție! Pentru acest test se rezolvă doar cerința 2.

(Olimpiada județeană, 2015)

4. SUBPROGRAME

1. De la tastatură se citește un număr întreg n . Să se scrie o funcție prin intermediul căreia se va determina numărul de cifre ale lui n .
2. De la tastatură se citește un număr întreg n . Să se scrie o funcție prin intermediul căreia se va determina numărul de cifre pare ale lui n .
- ✓ 3. De la tastatură se citește un număr întreg n . Să se scrie o funcție care să determine cifra maximă a lui n .
4. De la tastatură se citește un număr întreg n . Să se scrie un subprogram care va afișa cifrele care se repetă de cel puțin 2 ori.
5. De la tastatură se citește un număr întreg n . Folosind un subprogram, să se determine divizorii numărului n .
6. Se consideră subprogramul *Divizori*, cu trei parametri: n , prin care primește o valoare naturală ($0 < n < 10^6$); *mic* și *mare*, prin care furnizează cel mai mic și respectiv cel mai mare divizor propriu pozitiv al lui n (diferit de 1 și de n) sau valoarea 0, prin amândoi, dacă numărul nu are cel puțin doi astfel de divizori. Scrieți definiția completă a subprogramului. (Examen de Bacalaureat – Simulare 2014)
7. De la tastatură se citește un număr întreg n . Să se scrie o funcție prin intermediul căreia se va determina cel mai mare număr care se poate forma din cifrele lui n .
- ✓ 8. Se citesc de la tastatură 2 numere naturale a și b . Să se scrie o funcție care determină *cmmdc* a celor 2 numere.
- ✓ 9. Se citesc de la tastatură 2 numere naturale a și b . Să se scrie o funcție care determină *cmmmc* a celor 2 numere.
10. Se citesc de la tastatură 2 numere naturale a și b . Să se scrie un subprogram care determină cifrele comune celor 2 numere.
11. Se citesc de la tastatură 2 numere naturale a și b . Să se scrie o funcție care determină câte cifre din primul număr nu se regăsesc în al doilea număr.
- ✓ 12. Să se scrie o funcție care să returneze pentru un număr natural n transmis ca parametru valoarea lui $n!$ ($n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$).
- ✓ 13. Un interval cu proprietatea că există un singur număr natural, n ($2 \leq n$), pentru care valoarea produsului $1 \cdot 2 \cdot 3 \cdot \dots \cdot n$ aparține acestui interval este numit *interval factorial* al lui n . Exemplu: [5,8] și [3,23] sunt intervale factoriale ale lui 3, dar [1,15] și [7,10] nu sunt intervale factoriale ale nici unui număr. Se consideră subprogramul *Interval*, cu trei parametri: n , prin care primește un număr natural din intervalul [2,10], a și b , prin care furnizează câte un număr natural, astfel încât expresia $b - a$ să aibă valoare maximă, iar $[a, b]$ să fie interval factorial al lui n . Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - Iunie 2014)

- ✓ 14. Să se scrie o funcție care să returneze suma cifrelor unui număr natural transmis ca parametru.
- ✓ 15. Să se scrie o funcție care să returneze oglinditul unui număr natural transmis ca parametru.
16. Să se scrie o funcție care verifică dacă un număr natural transmis ca parametru este prim.
- ✓ 17. Să se scrie o funcție care să determine suma divizorilor unui număr natural transmis ca parametru.
- ✓ 18. Se consideră subprogramul *Triplete*, cu un singur parametru, n , prin care primește o valoare naturală din intervalul $[2, 10^4]$. Subprogramul afișează pe ecran toate tripletele de numere naturale (x, y, z) cu proprietatea că $x < y < z$ și $x \cdot y + y \cdot z = n$. Fiecare triplet se afișează pe câte o linie a ecranului, iar numerele din fiecare triplet sunt separate prin câte o virgulă și încadrate între paranteze rotunde. Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - August 2014)
19. Să se scrie o funcție care să returneze numărul de cifre diferite de zero ale unui număr natural transmis ca parametru.
20. Să se scrie o funcție care să returneze pentru un număr natural n transmis ca parametru numărul de cifre zero de la finalul lui n !
21. Scrieți definiția completă a unui subprogram *Nr_prim* care primește prin singurul său parametru n , un număr natural cu cel mult 9 cifre și returnează cel mai mic număr prim, strict mai mare decât n . (Variante Bacalaureat 2009)
22. Să se scrie o funcție care primește doi parametri x și y și returnează numărul obținut prin concatenarea lui x cu y .
23. Subprogramul *minDivPrim* are un singur parametru n , prin care primește un număr natural. Subprogramul returnează cel mai mic număr natural care are aceiași divizori primi ca n . (Variante Bacalaureat 2009)
24. Scrieți o funcție pentru a verifica dacă un număr este palindrom sau nu.
25. Să se afișeze toate numerele prime din intervalul $[a, b]$. Se va folosi o funcție pentru a verifica dacă un număr este prim.
26. Fie v un vector cu n elemente numere întregi. Să se scrie un program care determină suma elementelor numere prime conținute de vector. Se va folosi o funcție pentru a verifica dacă un număr este prim sau nu.
27. Să se afișeze numerele din intervalul $[a, b]$ care se divid cu suma cifrelor lor.
28. Să se afișeze numărul (numerele) din intervalul $[a, b]$ dat, care are cei mai mulți divizori. Se va folosi o funcție care calculează numărul de divizori ai unui număr.
29. Să se afișeze primele n numere pentru care suma cifrelor lor este un număr pătrat perfect. Se va folosi o funcție pentru calculul sumei cifrelor.

30. Subprogramul *CifreImpare* are un singur parametru, n , prin care primește un număr natural cu toate cifrele nenule ($n \in [1, 10^9]$). Subprogramul returnează numărul obținut prin eliminarea tuturor cifrelor impare din n , respectiv -1 dacă nu există astfel de cifre sau dacă toate cifrele lui n sunt impare. Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - Iunie 2016)
31. Să se scrie o funcție *Putere* care primește prin intermediul parametrului n un număr natural și returnează prin intermediul parametrilor x și k două numere naturale cu proprietatea că $n = x^k$, x este minim.
- ✓ 32. Să se scrie o funcție care primește ca parametru un număr natural n și returnează 1 dacă oricare două cifre alăturate din n au parități diferite și 0 în caz contrar.
- ✓ 33. Să se scrie o funcție care determină a k -a cifră a unui număr n dat. Numerotarea se face de la dreapta începând cu cifra unităților, poziția 1.
34. Scrieți definiția completă a funcției *Multipli*, cu trei parametri a, b, x ($a \leq b$), numere naturale, ce returnează numărul multiplilor lui x din intervalul $[a, b]$.
35. Subprogramul *Duplicare* are doi parametri: n , prin care primește un număr natural ($n \in [1, 10^4]$) și d , prin care furnizează numărul obținut prin duplicarea fiecărei cifre pare a lui n sau -1 dacă acesta nu are nici o cifră pară. Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - August 2016)
36. Subprogramul *Duplicare* are un singur parametru, n , prin care primește un număr natural ($n \in [1, 10^4]$). Subprogramul furnizează, prin același parametru, numărul obținut din n prin inserarea, după fiecare cifră pară din scrierea lui, a unei cifre egale cu jumătate din valoarea acesteia. Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - Iunie 2017)
37. Subprogramul *triplete* are trei parametri, a, b și c , prin care primește câte un număr natural din intervalul $[0, 10^4]$ ($a < b$). Subprogramul afișează pe ecran toate tripletele de numere naturale, x, y și z , soluții ale ecuației $x + y + z = c$, cu proprietatea că $a \leq x \leq y \leq z \leq b$. Fiecare triplet afișat este încadrat între acolade, iar numerele x, y și z sunt afișate în această ordine, separate prin câte o virgulă. Dacă nu există nici un triplet cu proprietatea cerută, se afișează mesajul *Nu există*. Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - August 2017)
- ✓ 38. Să se scrie o funcție care să determine prima și ultima cifră a unui număr natural transmis ca parametru. Funcția va întoarce rezultatele prin intermediul unor parametri de ieșire.
- ✓ 39. Subprogramul *NrPrime* are un singur parametru, n , prin care primește un număr natural ($n \in [0, 10^9]$). Subprogramul returnează numărul de cifre prime ale lui n . Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - August 2015)

40. Subprogramul *Identice* are doi parametri, a și b , prin care primește câte un număr natural ($10 \leq a \leq b \leq 10^6$). Subprogramul afișează pe ecran toate numerele naturale din intervalul $[a, b]$ care au toate cifrele identice. Numerele afișate sunt separate prin câte un spațiu, iar dacă nu există astfel de numere, se afișează pe ecran mesajul *Nu există*. Scrieți definiția completă a subprogramului. (Examen de Bacalaureat – Sesiunea Specială 2017)
- ✓ 41. Să se determine toate numerele perfecte mai mici sau egale decât n (n citit de la tastatură). Se va folosi o funcție pentru a determina dacă un număr este *perfect* (un număr este *perfect* dacă este egal cu suma divizorilor săi fără el însuși).
- ✓ 42. Să se scrie o funcție care să determine numărul divizorilor impari ai unui număr natural transmis ca parametru. Funcția întoarce rezultatul prin intermediul unui parametru de ieșire.
- ✓ 43. Scrieți definiția completă a funcției *Sub* care primește doi parametri n și k și determină afișarea pe ecran, în ordine descrescătoare, a primelor n numere naturale nenule divizibile cu k . (Variante Bacalaureat 2009)
- ✓ 44. Subprogramul *nrDiv* are doi parametri, a și b ($a \leq b$), prin care primește câte un număr natural din intervalul $[1, 10^9]$. Subprogramul returnează numărul valorilor din intervalul $[a, b]$ care pot fi scrise ca produs de două numere naturale consecutive. Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - Model 2017)
- ✓ 45. Subprogramul *DivImpar* are doi parametri, a și b , prin care primește două numere naturale din intervalul $[1, 10^5]$. Subprogramul returnează cel mai mare divizor comun impar al numerelor a și b . Scrieți definiția completă a subprogramului. (Examen de Bacalaureat – Sesiunea Specială 2015)
46. Numerele *puternice* sunt acele numere care au numărul divizorilor mai mare decât numărul divizorilor oricărui număr natural mai mic decât numărul respectiv. Să se afișeze al n -lea număr puternic. Se va folosi o funcție pentru a verifica dacă un număr este puternic sau nu.
- ✓ 47. Un număr natural nenul se numește *sPar* dacă atât el, cât și suma divizorilor săi proprii (divizori diferiți de 1 și de el însuși), sunt numere pare. Subprogramul *sPar* are un singur parametru, n , prin care primește un număr natural ($n \in [1, 10^9]$). Subprogramul returnează cel mai mic număr *sPar*, strict mai mare decât n . Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - Simulare 2016)
- ✓ 48. Un număr natural nenul se numește *subperfect* dacă este strict mai mic decât suma divizorilor săi proprii (divizori naturali diferiți de 1 și de el însuși). Exemplu: 12 este număr subperfect pentru că $12 < 2 + 3 + 4 + 6$. Se consideră subprogramul subperfect, cu doi parametri, a și b , prin care primește câte un număr natural ($2 \leq a \leq b \leq 10^9$). Subprogramul afișează pe ecran, separate prin câte un spațiu, în ordine descrescătoare, toate numerele subperfecte din intervalul $[a, b]$. Dacă în interval nu există astfel de numere, subprogramul afișează pe ecran mesajul nu exista. Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - Model 2015)
49. Se consideră doi vectori u și v cu m , respectiv n elemente. Să se determine reuniunea celor 2 vectori.
50. Fie v un vector cu n elemente numere întregi. Să se determine cel de-al k -lea element din vectorul ordonat crescător, fără a ordona vectorul.
- ✓ 51. Fie v un vector cu n elemente numere întregi. Să se determine poziția elementului minim din vector.
- ✓ 52. Scrieți definiția completă unui subprogram care returnează suma elementelor divizibile cu 3 ale unui tablou unidimensional transmis ca parametru. (Variante Bacalaureat 2009)
53. Scrieți definiția completă a subprogramului *Aranjare*, care are doi parametri v și n , prin care primește un tablou unidimensional cu maximum 10000 de numere naturale nenule și, respectiv, numărul de elemente din tablou. Subprogramul rearanjează elementele tabloului astfel încât toate valorile impare să se afle pe primele poziții, iar valorile pare în continuarea celor impare. (Variante Bacalaureat 2009)
- ✓ 54. Scrieți definiția completă a subprogramului *Multiplu* care are 3 parametri: a , prin care primește un tablou unidimensional cu maximum 100 de numere naturale mai mici decât 1000, n , numărul efectiv de elemente ale tabloului și k , un număr natural. Subprogramul returnează numărul de elemente din tablou care sunt multipli ai numărului k și au ultima cifră egală cu k . (Variante Bacalaureat 2009)
- ✓ 55. Scrieți definiția completă a subprogramului *Shift* care primește prin intermediul parametrului n o valoare naturală nenulă ($n \leq 100$), iar prin intermediul parametrului x , un tablou unidimensional cu n componente. Fiecare componentă a acestui tablou este un număr întreg care are cel mult 8 cifre. Subprogramul permută circular cu o poziție spre stânga primele n elemente ale tabloului x și furnizează tabloul modificat tot prin parametrul x . (Variante Bacalaureat 2009)
- ✓ 56. Scrieți definiția completă a subprogramului *Pminim* care primește prin intermediul parametrului n un număr natural nenul ($n \leq 100$), iar prin intermediul parametrului x un tablou unidimensional cu n componente întregi, de maximum opt cifre fiecare. Subprogramul furnizează prin intermediul parametrului *minv* valoarea minimă din tabloul x , prin intermediul parametrului *maxv* valoarea maximă din x , iar prin intermediul parametrului *sum* suma elementelor din tabloul x . (Variante Bacalaureat 2009)
- ✓ 57. Scrieți definiția completă a subprogramului *F*, care primește prin intermediul parametrului n un număr natural nenul ($1 \leq n \leq 9$), iar prin intermediul parametrului a , un tablou unidimensional care conține n valori naturale, fiecare dintre acestea reprezentând câte o cifră a unui număr. Astfel, a_0 reprezintă prima cifră a numărului, a_1 a doua cifră etc. Subprogramul furnizează prin parametrul k o valoare naturală egală cu numărul obținut din cifrele pare reținute în tabloul a sau valoarea -1 dacă în tablou nu există nici o cifră pară. (Variante Bacalaureat 2009)

- ✓ 58. Scrieți definiția completă a unui subprogram care returnează suma celor mai mici m elemente ale unui tablou unidimensional dat.
- ✓ 59. Fiind dat un vector v cu n elemente numere întregi, să se scrie un program care modifică acest vector, astfel încât el să conțină decât valori distincte. Se va folosi o funcție pentru a verifica dacă o valoare este deja în vector.
- ✓ 60. Fie v un vector cu n elemente numere întregi. Să se afișeze toate perechile de numere din vector care sunt prime între ele. Pentru calculul *cmmdc* se va folosi o funcție.
61. Scrieți definiția completă a subprogramului *Număr*, care primește prin intermediul parametrului n un număr natural nenul ($1 \leq n \leq 100$), iar prin intermediul parametrului a , un tablou unidimensional care conține n valori naturale.
Subprogramul furnizează prin parametrul k o valoare naturală egală cu numărul obținut prin concatenarea valorii maxime cu valoarea minimă din tablou. (Variante Bacalaureat 2009)
62. Se consideră doi vectori A cu m elemente și B cu n elemente. Se cere să se determine reuniunea, intersecția și diferența celor doi vectori. Pentru determinarea cerinței se vor folosi subprograme care vor avea ca parametri de intrare dimensiunile vectorilor.
63. Se consideră doi vectori A cu m elemente și B cu n elemente. Se cere să se determine reuniunea celor doi vectori folosind un subprogram pentru ordonarea unui vector și un subprogram pentru a insera o valoare într-un vector ordonat, pentru a rămâne ordonat (se va folosi căutarea binară a poziției de inserare).
64. Se citește de la tastatură un vector cu n elemente numere naturale. Să se ordoneze crescător elementele pătrate perfecte, plasându-le la începutul vectorului și apoi cele care nu sunt pătrate perfecte, plasându-le la sfârșitul vectorului. Să se afișeze vectorul astfel obținut. Pentru realizarea cerinței se vor folosi subprograme.
- ✓ 65. Scrieți definiția completă a unui subprogram care realizează ștergerea dintr-un tablou unidimensional dat a elementelor cu indici între două valori date a și b .
- ✓ 66. Se citește un vector v cu n elemente numere naturale. Să se verifice dacă elementele vectorului alternează ca paritate, adică oricare două elemente alăturate au parități diferite. Pentru verificarea alternării parității a două numere se va folosi o funcție.
67. Scrieți definiția completă a subprogramului *Interval* care are 2 parametri: a , prin care primește un tablou unidimensional cu maximum 100 de numere naturale mai mici decât 1000 și n , numărul efectiv de elemente ale tabloului.
Subprogramul returnează numărul de elemente din tablou care aparțin intervalului închis determinat de primul și ultimul element al tabloului.

- ✓ 68. Scrieți definiția completă a subprogramului *Count* care are 2 parametri: a , prin care primește un tablou unidimensional cu maximum 100 de numere reale și n , numărul efectiv de elemente ale tabloului. Subprogramul returnează numărul de elemente din tablou care sunt mai mari sau cel puțin egale cu media aritmetică a tuturor elementelor din tablou.
69. Scrieți definiția completă a subprogramului *Valuri* care are 2 parametri: n – prin care primește un număr natural, v , prin care furnizează un tablou unidimensional cu $2n$ elemente, valori naturale distincte din intervalul $[1, 2n]$.
Subprogramul construiește tabloul v astfel încât, în acesta, șirul elementelor impare să fie strict crescător, iar șirul elementelor pare să fie strict descrescător. Primul element al tabloului este impar, iar două elemente cu aceeași paritate nu pot ocupa poziții consecutive în tablou. (Examen de Bacalaureat - Iunie 2013)
70. Se consideră subprogramul *Sub*, cu trei parametri: n , prin care primește un număr natural ($2 < n < 50$); v , prin care primește un tablou unidimensional cu n elemente, numere naturale cu cel mult 4 cifre; k , prin care primește un număr natural ($1 < k \leq n$). Subprogramul returnează suma primelor k elemente cu valoare impară ale tabloului. Dacă nu există k elemente impare în tablou, subprogramul returnează valoarea -1. Scrieți definiția completă a subprogramului. (Examen de Bacalaureat - August 2013)
71. Se citește de la tastatură o matrice A cu $m \times n$ elemente numere întregi. Să se determine linia (liniile) din matrice care conține cele mai multe elemente nule. Se va folosi o funcție care returnează numărul elementelor nule de pe o linie a cărui indice se transmite ca parametru.
72. Se dă o matrice A cu $m \times n$ elemente. Să se permute liniile matricei circular spre dreapta cu k poziții (k este dat). Se va folosi un subprogram care permută circular spre dreapta cu o poziție o linie a matricei.
73. Numim *linii gemene* într-o matrice, două linii care conțin aceleași elemente, eventual dispuse altfel pe coloane. Scrieți un program care determină toate perechile de linii gemene dintr-o matrice A dată, cu $m \times n$ elemente numere întregi. Pentru fiecare astfel de pereche se vor afișa indicii liniilor componente.
74. Se citește de la tastatură o matrice A cu m linii și n coloane. Să se afișeze indicii (indicii) coloanei care are cele mai multe pătrate perfecte în componența sa. Se va folosi o funcție care să găsească numerele pătrate perfecte pe o coloană a matricei.

75. Se consideră o matrice A cu m linii și n coloane cu elemente de 0 și 1 . Să se scrie un program care să afișeze toate perechile de linii complementare. Două linii se numesc *complementare* dacă suma componentelor de pe aceeași coloană este 1 . Pentru a verifica dacă două linii sunt complementare se va folosi o funcție.

76. Se consideră o matrice a cu n linii și n coloane și elemente numere întregi. Valoarea lui n este un număr impar. Putem împărți o astfel de matrice în patru cadrane, prin intermediul liniei și coloanei din mijloc. Să se calculeze și să se afișeze sumele din fiecare cadran. Sumele cerute nu vor conține și valorile de pe linia și coloana din mijloc. Pentru calculul unei astfel de sume se va folosi o funcție care va avea ca parametri indicii pentru colțul stânga-sus, respectiv dreapta-jos al fiecărui cadran.

77. Se citește de la tastatură o matrice A cu n linii și n coloane cu elemente numere întregi. Să se calculeze A^m (m este un număr natural dat, $m > 2$). Se va folosi un subprogram pentru calculul produsului a două matrice.

78. Se consideră un tablou bidimensional cu $n \times m$ elemente numere întregi. Să se creeze un program care ordonează crescător elementele situate pe liniile cu număr de ordine par (linia a doua, linia a patra etc). La afișare, elementele tabloului vor fi scrise sub forma unui tablou cu n linii și m coloane. Pentru ordonarea elementelor de pe o linie se va folosi un subprogram.

79. Se consideră o matrice pătratică a cu n linii și n coloane cu elemente numere întregi. Să se rearanjeze elementele matricei, astfel încât elementele de pe fiecare coloană să fie ordonate crescător. Pentru ordonarea elementelor de pe o coloană se va folosi un subprogram.

80. Se consideră o matrice cu n linii și n coloane și elemente numere naturale. Să se determine suma elementelor de pe diagonala principală și de pe semidiagonalele vecine cu diagonala secundară (din stânga și din dreapta). Pentru calculul sumei unei diagonale se va folosi o funcție.

5. RECURSIVITATE

5.1. Subprograme recursive

1. Să se scrie o funcție recursivă care returnează factorialul unui număr dat ca parametru.
2. Să se scrie un subprogram recursiv care determină factorialul unui număr transmis ca parametru și întoarce rezultatul prin intermediul unui parametru de ieșire.
3. Să se scrie o funcție recursivă care să returneze suma cifrelor unui număr natural transmis ca parametru.
4. Să se scrie o funcție recursivă care să returneze câte cifre conține un număr natural transmis ca parametru.
5. Să se scrie un subprogram recursiv care determină suma cifrelor unui număr natural n transmis ca parametru și întoarce rezultatul prin intermediul unui parametru de ieșire.
6. Să se scrie o funcție recursivă care să returneze numărul de cifre diferite de zero ale unui număr natural transmis ca parametru.
7. Să se scrie un subprogram recursiv care să determine numărul de cifre diferite de zero ale unui număr natural transmis ca parametru și să întoarcă rezultatul prin intermediul unui parametru de ieșire.
8. Să se scrie un subprogram recursiv care returnează reprezentarea unui număr n dat (n este în baza 10) într-o bază b dată.
9. Se citește de la tastatură un număr natural n . Să se scrie un subprogram recursiv care să returneze valoarea sumei $1 + 3 + 5 + \dots + 2n - 1$.
10. Se citește de la tastatură un număr natural n . Să se scrie un subprogram recursiv care să returneze valoarea sumei $2 + 4 + 6 + \dots + 2n$.
11. Se citește de la tastatură un număr natural n . Să se scrie un subprogram recursiv care să returneze valoarea sumei $2 + 5 + 8 + \dots + 3n - 1$.
12. Se citește de la tastatură un număr natural n . Să se scrie un subprogram recursiv care să returneze valoarea sumei $1 + 5 + 9 + \dots + 4n - 3$.
13. Se citește de la tastatură un număr natural n . Să se scrie un subprogram recursiv care să returneze valoarea sumei $\frac{1}{2} + \frac{1}{7} + \frac{1}{12} + \dots + \frac{1}{5n-3}$.
14. Se citește de la tastatură un număr natural n . Să se scrie un subprogram recursiv care să returneze valoarea sumei $\frac{1}{3} + \frac{1}{8} + \frac{1}{13} + \dots + \frac{1}{5n-2}$.
15. Se citește de la tastatură un număr natural n . Să se scrie o funcție recursivă pentru calculul produsului $1 \cdot 4 \cdot 7 \cdot \dots \cdot (3n - 2)$.
16. Se citește de la tastatură un număr natural n . Să se scrie o funcție recursivă pentru calculul expresiei $E = 1 \cdot 2 - 2 \cdot 4 + \dots + (-1)^{n+1} \cdot n \cdot 2n$.

17. Să se scrie o funcție recursivă care să returneze cifra maximă a unui număr natural transmis ca parametru.
18. Să se scrie o funcție recursivă care să returneze cifra minimă a unui număr natural transmis ca parametru.
19. Să se scrie un subprogram recursiv care să returneze cea mai mică cifră pară a unui număr natural transmis ca parametru. Rezultatul va fi returnat tot printr-un parametru.
20. Să se scrie o funcție recursivă care verifică dacă un număr întreg este prim sau nu.
21. Scrieți un program care afișează toate numerele prime mai mici decât n (n este dat).
22. Folosind recursivitatea, să se verifice dacă un număr n este palindrom (numărul este egal cu oglinditul său).
23. Scrieți o funcție recursivă care calculează suma divizorilor unui număr n dat.
24. Scrieți o funcție recursivă care testează dacă un număr natural este bine ordonat descrescător (cifrele sale citite de la stânga la dreapta sunt în ordine descrescătoare).
25. Să se scrie o funcție recursivă care returnează cel mai mare divizor comun a două numere transmise ca parametri.
26. Să se scrie un subprogram recursiv care determină cel mai mare divizor comun a două numere transmise ca parametri și întoarce rezultatul prin intermediul unui parametru de ieșire.
27. Să se afișeze toate numerele naturale mai mici decât n care sunt egale cu suma factorialilor lor.
28. Să se afișeze în ordine crescătoare toate numerele de câte n cifre ($0 < n < 10$) cu suma cifrelor egală cu $5n$.
29. Să se scrie o funcție recursivă pentru a înmulți două numere întregi, folosind adunări repetate.
30. Să se scrie o funcție recursivă pentru a împărți două numere întregi folosind scăderi repetate.
31. O persoană are de coborât n trepte. La fiecare pas ea poate coborî 1 sau 2 trepte. Să se determine câte posibilități are persoana de a coborî scara.
32. Pentru un întreg n dat, să se construiască recursiv triunghiul de numere de mai jos, fără a folosi cicluri:

```

n  n - 1 ... 3 2 1
n - 1 n - 2 ... 2 1
...
1

```

33. Pentru un întreg n dat, să se construiască triunghiul de numere de mai jos, fără a folosi cicluri:

```

n  n - 1 ..... 3 2 1
n  n - 1 ..... 3 2
n  n - 1 ..... 3
...
n

```

34. Să se scrie un suprogram care să determine al n -lea termen al șirului lui Fibonacci, dat prin formula de recurență:

$$f_n = \begin{cases} 1, & \text{dacă } n = 0 \text{ sau } n = 1 \\ f_{n-1} + f_{n-2}, & \text{altfel} \end{cases}$$

35. Se consideră funcția $Ack: N \times N \rightarrow N$ dată prin următoarea formulă de recurență

$$Ack(m, n) = \begin{cases} n + 1, & m = 0 \\ Ack(m - 1, 1), & n = 0 \\ Ack(m - 1, Ack(m, n - 1)), & \text{altfel} \end{cases}$$

Pentru valorile m și n citite de la tastatură, să se afișeze $Ack(m, n)$.

36. Să se realizeze un program pentru calculul și afișarea valorilor calculate pentru *triunghiul lui Pascal* până la nivelul k (k este dat). (Triunghiul lui Pascal este format din coeficienții *binomului lui Newton* $(a + b)^n$ pentru $n = 1, 2, \dots, k$.)
37. Să se genereze primii n termeni ai unei progresii aritmetice dacă se cunosc primul termen și rația.
38. Să se scrie un program care descompune un număr n în factori primi, folosind un subprogram recursiv.
39. Să se scrie un program care descompune un număr dat n ca sumă de puteri distincte ale lui 2.
40. Un număr *super-prim* este un număr ale cărui sufixe sunt tot numere prime. De exemplu 239 este număr super prim. Să se scrie un program care, folosind suprograme recursive, să afișeze numerele super-prime mai mici sau egale cu o valoare dată n .
41. *Triunghiul lui Sierpinski*. Se dă un triunghi oarecare, prin coordonatele vârfurilor triunghiului. Se împarte triunghiul în patru triunghiuri de arie egală ducând cele trei linii mijlocii ale triunghiului. Se repetă construcția pentru triunghiurile mici formate, cu excepția triunghiului format doar din linii mijlocii (cel din interior). Se repetă procedeul de o infinitate de ori. Problema pusă de Sierpinski este de a determina aria suprafeței rămase după eliminarea triunghiurilor construite la fiecare pas numai din linii mijlocii. Să se illustreze problema lui Sierpinski prin aplicarea de n ori a procedeului.

5.2. Recursivitate cu vectori

1. Se citește de la tastatură un vector v cu n elemente numere naturale. Să se determine, folosind o funcție recursivă numărul componentelor negative din vector.
2. Se citește de la tastatură un vector v cu n elemente numere naturale. Să se afișeze, folosind un subprogram recursiv, pozițiile elementelor pozitive din vector.
3. Se citește de la tastatură un vector cu n elemente numere întregi. Să se scrie o funcție recursivă care returnează maximul dintre valorile negative, aflate pe poziții pare în cadrul vectorului.
4. Să se scrie o funcție recursivă care afișează minimul dintre elementele pozitive ale unui șir de n elemente numere întregi.
5. Se consideră un vector x cu n elemente numere întregi. Să se realizeze un program recursiv care calculează produsul elementelor de pe pozițiile impare din vector. Pozițiile în vector sunt numerotate începând cu poziția 1.
6. Se citește de la tastatură un vector cu n elemente numere întregi. Să se scrie o funcție recursivă care returnează de câte ori apare elementul minim în cadrul vectorului.
7. Scrieți definiția completă a unui subprogram recursiv care returnează suma elementelor unui tablou unidimensional transmis ca parametru. (Variante Bacalaureat 2009)
8. Scrieți definiția completă unui subprogram recursiv care returnează suma elementelor divizibile cu 3 ale unui tablou unidimensional transmis ca parametru. (Variante Bacalaureat 2009)
9. Să se scrie un subprogram recursiv care afișează pe ecran elementele unui vector transmis ca parametru.
10. Se citește de la tastatură un șir de n numere întregi. Să se scrie o funcție recursivă care returnează numărul schimbării de semn între două elemente consecutive ale șirului.
11. Se citește un vector cu n elemente numere naturale ($n \leq 100$). Să se înlocuiască fiecare element al vectorului cu suma cifrelor de aceeași paritate cu poziția elementului. Prima poziție în vector este poziția 1.
12. Să se scrie un subprogram recursiv care citește de la tastatură un șir de valori naturale și le afișează în ordinea inversă introducerii lor.
13. Să se verifice dacă elementele unui vector cu n elemente numere întregi sunt în ordine strict crescătoare. Se va folosi o funcție care va returna rezultatul logic al testării.
14. Se citește de la tastatură un vector v cu n elemente numere naturale. Să se determine, folosind o funcție recursivă suma numerelor prime din vector.
15. Se dă de la tastatură un vector v cu n elemente numere naturale. Să se afișeze componentele din vector care au suma cifrelor divizibilă cu k (k este dat). Se va folosi o funcție recursivă pentru calculul sumei cifrelor.
16. Să se verifice, folosind o funcție recursivă, dacă un șir de numere are toate elementele pozitive.
17. Să se scrie o funcție recursivă care returnează media aritmetică a elementelor numere prime dintr-un vector v cu n elemente numere întregi. Pentru testarea proprietății de număr prim se va folosi o altă funcție.
18. Să se scrie un program, care, folosind funcții recursive, calculează și afișează suma pătratelor perfecte dintr-un vector v cu n elemente numere naturale (un număr este pătrat perfect dacă radicalul său este un număr natural).
19. Se citește de la tastatură un vector v cu n elemente numere naturale. Să se scrie o funcție recursivă care să returneze numărul divizorilor primi ai unui număr natural x , transmis ca parametru, apoi, să se afișeze numărul cerut pentru fiecare dintre elementele vectorului dat.
20. Se citește un vector v cu n elemente numere naturale. Să se scrie un program care, folosind un subprogram recursiv determină elementul din șir, cel mai apropiat de un pătrat perfect.
21. Se citește de la tastatură un șir de n numere naturale. Folosind un algoritm recursiv, să se afișeze toate elementele șirului cu număr maxim de cifre distincte.
22. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se scrie un subprogram, care, parcurgând recursiv vectorul, afișează elementele acestuia care au suma cifrelor maximă. Pentru calculul sumei cifrelor se va folosi o funcție recursivă.
23. Scrieți un program care verifică dacă un șir cu n elemente este ordonat crescător.
24. Folosind un subprogram recursiv, să se afle maximul dintre elementele unui vector cu n elemente (n este dat).
25. Folosind o funcție recursivă, să se verifice dacă o valoare dată b se află într-un vector cu n elemente numere întregi. Elementele vectorului, precum și numărul lor se citesc de la tastatură.
26. Se consideră un șir x cu n elemente numere întregi. Să se realizeze un subprogram recursiv care construiește un șir y ce va conține toate elementele pozitive din șirul x .
27. Se consideră un șir x cu n elemente numere întregi. Să se realizeze un subprogram recursiv care construiește un șir y ce va conține pozițiile elementelor pare din șirul x .
28. Scrieți o funcție recursivă care returnează produsul scalar a doi vectori u și v cu câte n elemente fiecare: $u_1 \times v_1 + u_2 \times v_2 + \dots + u_n \times v_n$.
29. Scrieți o variantă recursivă a algoritmului de sortare prin inserție.
30. Se citește de la tastatură un șir de n numere naturale. Să se realizeze un subprogram recursiv care afișează perechile de numere consecutive din șir cu proprietatea că cel mai mare divizor comun al elementelor perechii este egal cu o valoare dată d .

5.3. Recursivitate cu matrice

1. Se citește de la tastatură o matrice a cu m linii și n coloane cu elemente numere întregi. Să se scrie un subprogram recursiv pentru afișarea elementelor pozitive din matricea a .
2. Se citește de la tastatură o matrice a cu m linii și n coloane cu elemente numere întregi. Să se scrie o funcție recursivă care să returneze suma elementelor din matricea a .
3. Se citește de la tastatură o matrice a cu m linii și n coloane cu elemente numere întregi. Să se scrie o funcție recursivă care să returneze produsul elementelor negative din matricea a .
4. Se citește de la tastatură o matrice a cu m linii și n coloane cu elemente numere întregi. Să se realizeze o funcție recursivă care să returneze minimul elementelor pozitive din matricea a .
5. Se citește de la tastatură o matrice a cu m linii și n coloane cu elemente numere întregi. Să se scrie un subprogram recursiv pentru calculul sumei elementelor de pe o linie l a matricei (l este dat de la tastatură).
6. Se citește de la tastatură o matrice a cu m linii și n coloane cu elemente numere întregi. Să se scrie un subprogram recursiv pentru calculul sumei elementelor de pe o coloană dată c a matricei.
7. Se citește de la tastatură o matrice a cu n linii și n coloane cu elemente numere întregi. Folosind o funcție recursivă să se calculeze și să se afișeze suma elementelor de pe diagonala principală a matricei.
8. Se citește de la tastatură o matrice a cu n linii și n coloane cu elemente numere întregi. Folosind o funcție recursivă să se calculeze și să se afișeze suma elementelor de deasupra diagonalei principale a matricei.
9. Se citește de la tastatură o matrice a cu n linii și n coloane cu elemente numere întregi. Folosind o funcție recursivă să se calculeze și să se afișeze suma elementelor de sub diagonala secundară a matricei.
10. Se citește de la tastatură o matrice a cu n linii și n coloane cu elemente numere întregi. Să se scrie un program care, folosind un algoritm recursiv, scrie în fișierul "*Diagonala.txt*", pe un singur rând, separate prin spații, elementele pozitive aflate sub diagonala principală în matrice.
11. Să se afișeze elementele unei matrice A cu $n \times n$ elemente numere întregi prin parcurgerea ei în spirală. Matricea se va citi din fișierul "*Matrice.txt*".
12. Se citește de la tastatură un șir cu $n \times n$ elemente numere întregi. Să se introducă elementele șirului într-o matrice pătratică cu n linii și n coloane prin parcurgerea matricei în spirală. Parcurgerea laturilor ce alcătuiesc spirala se va face recursiv.
13. O matrice pătratică A poate fi împărțită de diagonale în 4 zone: *nord*, *sud*, *est*, *vest* (elementele de pe diagonală nu fac parte din nici o zonă). Să se scrie câte o funcție recursivă pentru a calcula:
 - a) suma elementelor din nord;
 - b) produsul elementelor din est.

Elementele matricei, precum și numărul de linii și coloane se vor citi din fișierul "*Matrice.txt*".

14. Să se completeze în spirală o matrice cu $n \times n$ elemente cu primele n^2 numere întregi de 2 cifre cu proprietatea că suma cifrelor este divizibilă cu 3.
15. Să se scrie un program, care folosind recursivitatea rotește cu 180° un tablou pătratic.

5.4. Recursivitate cu șiruri de caractere

1. Folosind recursivitatea, să se verifice dacă două șiruri de caractere s și t sunt anagrame (unul se poate obține din celălalt prin permutări de litere).
2. Să se scrie un program care afișează în ordine inversă citirii un șir de caractere, folosind recursivitatea. Introducerea se termină prin caracterul #.
3. Se citește de la tastatură, caracter cu caracter, un șir de caractere. Citirea se încheie la întâlnirea caracterului \$. Folosind un algoritm recursiv, să se afișeze în ordinea inversă citirii, cifrele care apar în șir.
4. Să se scrie o funcție recursivă care primește ca parametri un număr natural n și o cifră c și returnează numărul obținut prin eliminarea din n a tuturor aparițiilor lui c .
5. Folosind un subprogram recursiv, să se afișeze toate prefixele unui cuvânt dat, în ordinea descrescătoare a lungimii lor, pe linii separate.
6. Folosind subprogram recursiv să se afișeze toate sufixele unui cuvânt dat, în ordinea descrescătoare a lungimilor lor, pe linii separate.
7. Folosind un subprogram recursiv, să se afișeze toate secvențele mediane ale unui cuvânt dat.
8. Să se scrie o funcție recursivă care returnează numărul de vocale dintr-un șir de caractere transmis ca parametru.
9. Să se scrie o funcție recursivă care returnează numărul de consoane dintr-un șir de caractere transmis ca parametru.
10. Să se scrie o funcție recursivă care returnează numărul de cifre și caractere speciale dintr-un șir de caractere transmis ca parametru.
11. Fișierul "*Cuvinte.txt*" conține mai multe cuvinte, dispuse fiecare pe câte un rând. Să se scrie un subprogram recursiv care afișează cuvântul de lungime maximă.

5.5. Recursivitate indirectă

1. Să se determine valoarea funcției $f(x, y) = \begin{cases} x, & \text{pentru } x \geq y \\ g(x, y - 2), & \text{pentru } x < y \end{cases}$
Unde $g(x, y) = \begin{cases} x + y - 4, & \text{pentru } x + y \geq 4 \\ 2 \cdot f(0, y), & \text{pentru } x + y < 4 \end{cases}$
2. Fie șirul $a_0, b_0, a_1, b_1, a_2, b_2, \dots, a_n, b_n$, unde $a_n = \frac{a_{n-1} + b_{n-1}}{2}$ și $b_n = \sqrt{a_{n-1} b_{n-1}}$. Să se scrie un program care să calculeze a_n și b_n .

6. METODE DE PROGRAMARE

6.1. Divide et Impera

1. Folosind metoda *Divide et Impera*, să se afișeze elementele negative dintr-un șir cu n elemente numere întregi, date.
2. Se citește de la tastatură un vector cu n elemente numere întregi. Să se determine, folosind un algoritm *Divide et Impera*, numărul elementelor pare.
3. Se citește de la tastatură un vector cu n elemente numere întregi. Să se verifice, folosind un algoritm *Divide et Impera*, dacă toate elementele din vector sunt impare.
4. Se citește de la tastatură un vector cu n elemente numere întregi. Să se verifice dacă vectorul conține cel puțin un element par.
5. Se consideră un șir cu n elemente, numere naturale. Folosind metoda *Divide et Impera*, determinați suma elementelor acestui șir.
6. Se consideră un șir cu n elemente, numere naturale. Să se determine suma elementelor impare din acest șir.
7. Se dă un șir cu n elemente, numere naturale. Să se verifice dacă toate elementele șirului sunt pare.
8. Se citește de la tastatură un vector cu n elemente numere întregi. Să se verifice dacă vectorul este sortat crescător.
9. Se citește de la tastatură un vector v cu n elemente numere întregi. Să se construiască un vector u care să conțină elementele pozitive din vectorul v .
10. Să se determine simultan maximul și minimul dintr-un vector cu n elemente numere întregi.
11. Se dau n numere naturale. Să se afișeze al k -lea cel mai mic element din șir.
12. Se dă un șir cu n elemente, numere naturale. Să se verifice dacă în șir există elemente pozitive.
13. Se dă un șir cu n elemente, numere naturale. Să se verifice dacă în șir există elemente numere prime.
14. Se consideră un șir cu n elemente, numere naturale. Să se determine cel mai mare element prim din acest șir.
15. Se consideră un număr natural n . Să se calculeze $n!$, folosind un algoritm *Divide et Impera*.
16. Se consideră o matrice pătratică A cu n linii și n coloane, cu elemente numere întregi. Să se realizeze un subprogram, folosind metoda *Divide et Impera* pentru afișarea elementelor pozitive din matrice.
17. Se consideră o matrice pătratică A cu n linii și n coloane, cu elemente numere întregi. Să se scrie o funcție care returnează suma elementelor negative din matricea A .
18. Se consideră o matrice pătratică A cu n linii și n coloane, cu elemente numere întregi. Să se scrie o funcție care returnează produsul elementelor din matricea A .
19. Se consideră o matrice pătratică A cu n linii și n coloane, cu elemente numere întregi. Să se scrie o funcție care returnează maximul elementelor din matricea A .
20. Se consideră o matrice a cu n linii și n coloane. Să se afișeze elementele pozitive ale matricei.
21. Se consideră o matrice pătratică A cu n linii și n coloane, cu elemente numere întregi. Să se scrie o funcție care returnează minimul elementelor pozitive din matricea A .
22. Se consideră o matrice a cu n linii și n coloane. Să se afișeze elementele de deasupra diagonalei principale din matrice.
23. Se consideră o matrice a cu n linii și n coloane. Să se afișeze maximul elementelor din matrice.
24. Considerăm un șir de n numere întregi, ordonat crescător și un număr întreg x . Scrieți un algoritm care împarte șirul dat în două subșiruri în așa fel încât toate elementele primului subșir să fie mai mici sau cel mult egale cu x , iar toate elementele celui de-al doilea subșir să fie strict mai mari decât x . Elementele șirului și valoarea lui n se citesc din fișierul "Vector.in".
25. Se consideră un vector cu n elemente numere naturale. Definim *plierea vectorului*, ca fiind suprapunerea unei jumătăți numită donatoare peste cealaltă jumătate, numită receptoare. În cazul în care vectorul are un număr impar de elemente, cel din mijloc este eliminat. În acest mod se ajunge la un vector ale cărui elemente încep de la o poziție pi și se termină la o poziție pf . Plierea se poate aplica în continuare în mod repetat până când se ajunge la un subvector de lungime 1 ($pi = pf$) numit element final. Să se precizeze toate elementele finale posibile, precum și o serie de plieri prin care se poate ajunge la ele.
26. Pentru un șir de n numere întregi date $a_1, a_2, \dots, a_n$ să se găsească n numere întregi $x_1, x_2, \dots, x_n$, astfel încât $cmmdc$ al numerelor $a_1, a_2, \dots, a_n$ să se scrie sub forma: $x_1 \cdot a_1 + x_2 \cdot a_2 + \dots + x_n \cdot a_n$.
27. Folosind metoda *Divide et Impera*, să se implementeze un algoritm recursiv de înmulțire a n numere naturale date $x_1, x_2, \dots, x_n$. De exemplu, pentru $n = 5$, produsul $x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5$ se poate calcula sub forma $((x_1 \cdot x_2) \cdot x_3) \cdot (x_4 \cdot x_5)$.
28. Să se sorteze prin metoda rapidă un șir de n elemente numere întregi.
29. Să se ordoneze crescător un șir cu n numere, folosind sortarea prin interclasare.

30. Se dă un vector cu n elemente, numere naturale și un număr k . Ordonăți crescător primele k elemente ale vectorului și descrescător ultimele $n - k$ elemente.
31. Să se găsească *cmmdc* al unui șir de numere $x_1, x_2, \dots, x_n$.
32. Se consideră un șir de caractere. Să se inverseze caracterele din șir, folosind un subprogram recursiv care să folosească metoda *Divide et Impera*.
33. Fie un tablou unidimensional neordonat cu n numere întregi. Să se determine toate pozițiile în care apare în tablou o anumită valoare b , citită de la tastatură.
34. Se dă un vector cu n elemente numere naturale. Folosind metoda *Divide et Impera* să se precizeze câte elemente ale vectorului sunt egale.
35. Să se calculeze a^n unde a și n sunt numere naturale date.
36. Se dă un număr natural x . Să se scrie un algoritm care calculează partea întreagă din radicalul lui x . Nu se va folosi o funcție predefinită.
37. Să se calculeze valoarea unui polinom $P(x)$ într-un punct a dat.
38. Fie v un vector cu n elemente și k un număr întreg, $1 \leq k \leq n$. Să se găsească elementul care se află pe poziția k în șirul ordonat crescător, fără a efectua ordonarea.
39. Fie v un vector cu n elemente ordonate crescător. Să se determine un indice m , ($1 \leq m \leq n$), cu $v_m = m$, dacă este posibil.
40. *Turnurile din Hanoi* este un joc matematic. Este format din trei tije A, B și C și un număr variabil de discuri, de diferite diametre. Inițial discurile sunt așezate în ordine descrescătoare a diametrelor pe tija A , de la vârf către bază, astfel încât să formeze un turn.
Scopul jocului este acela de a muta toate discurile de pe tija A pe tija C folosind ca tijă intermediară tija B , respectând următoarele reguli:
- doar un singur disc poate fi mutat, la un moment dat;
- fiecare mutare constă în luarea celui mai de sus disc de pe o tijă și mutarea acestuia pe o altă tijă;
- un disc cu diametrul mai mare nu poate fi poziționat deasupra unui disc cu diametrul mai mic.
Dacă se cunoaște numărul n de discuri aflate pe tija A , să se determine șirul mutărilor necesare pentru ca toate discurile să fie mutate pe tija C .
41. Se consideră o bucată dreptunghiulară de tablă cu lungimea L și înălțimea H (L și H sunt 2 numere întregi date de la tastatură), pe care ne-o imaginăm așezată în primul cadran al unui sistem cartezian de coordonate, cu un colț al bucății de tablă în origine $O(0,0)$. Bucata de tablă are pe suprafața ei n găuri de coordonate întregi. Se cere să se decupeze din ea o bucată dreptunghiulară de arie maximă, care nu conține găuri. Diametrul unei găuri este suficient de mic astfel încât, în urma unei tăieturi, pe o direcție care conține o gaură se obțin margini drepte, nu ondulate. (*Problema tăieturilor*).

6.2. Backtracking

1. Se citește un număr natural nenul n . Să se afișeze, în ordine lexicografică, permutările mulțimii $\{1, 2, \dots, n\}$.
2. Să se genereze toate șirurile formate din n cifre, astfel încât fiecare șir generat să conțină cifre din mulțimea $\{2, 3, 4, 5\}$, iar fiecare două cifre alăturate sunt fie ambele pare, fie ambele impare.
3. Se citește un număr natural nenul n , apoi n numere naturale distincte. Să se afișeze, în ordine lexicografică, permutările mulțimii formate din cele n numere citite.
4. Se citește de la tastatură un număr natural n . Afișați permutările mulțimii $\{1, 2, \dots, n\}$ pentru care elementele până la jumătate vor fi în ordine crescătoare, iar de la jumătate până la final în ordine descrescătoare.
5. Fiind date m cărți, să se determine toate modalitățile de aranjare a acestora pe un raft (m este citit de la tastatură).
6. Se citesc două numere naturale nenule n și k . Să se afișeze, în ordine lexicografică, aranjamentele de câte k elemente ale mulțimii $\{1, 2, \dots, n\}$.
7. Fiind dată mulțimea $A = \{1, 4, 6, 7, 9\}$, să se genereze toate numerele formate din 3 cifre distincte care se pot forma cu elementele mulțimii date. Determinați numărul celor care se termină cu cifra 6.
8. Pentru a colora o hartă cu 5 țări se folosesc 7 culori. Să se genereze toate modalitățile de a colora hărțile și să se precizeze numărul lor.
9. Se citesc două numere naturale nenule n și k . Să se afișeze, în ordine lexicografică, submulțimile de câte k elemente ale mulțimii $\{1, 2, \dots, n\}$.
10. Se citesc două numere naturale n și k . Să se genereze toate numerele naturale formate din n cifre care conțin exact k cifre de 1.
11. Să se genereze toate numerele prime de n cifre ($n < 10$) formate numai cu ajutorul cifrelor c_1, c_2, c_3 citite de la tastatură.
12. Să se genereze toate numerele de cel mult n cifre, formate doar din cifre pare, cifre aflate în ordine strict crescătoare.
13. Să se determine toate numerele formate din n cifre distincte cu proprietatea că suma cifrelor este s .
14. Se citește un număr natural nenul n , apoi n numere naturale distincte. Să se afișeze, în ordine lexicografică, șirurile din cele n valori cu proprietatea că oricare două valori învecinate au parități diferite.
15. Să se genereze toate șirurile strict crescătoare formate din numere naturale, cu proprietatea că primul element din șir este egal cu n , iar ultimul element al șirului este cel mult egal cu $n + p$. Numerele n și p sunt citite de la tastatură.
16. Se citește un număr natural nenul n . Să se afișeze, în ordine lexicografică, toate submulțimile nevide ale mulțimii $\{1, 2, \dots, n\}$.

17. Se citește un număr natural nenul n . Să se afișeze, în ordine lexicografică, toate submulțimile nevide ale mulțimii $\{1, 2, \dots, n\}$ pentru care diferența dintre oricare două elemente este mai mare decât 2.
18. Se citesc două numere naturale nenule n și m . Să se determine toate șirurile cu m elemente din mulțimea $\{1, 2, \dots, n\}$, ordonate strict crescător, cu proprietatea că oricare două elemente consecutive în șir au diferența mai mică sau egală cu 2 .
19. Se dau două numere naturale nenule n și m . Considerăm mulțimea $A = \{1, 2, \dots, n\}$. Să se genereze în ordine lexicografică elementele produsului cartezian $A^m = A \times A \times \dots \times A$.
20. Se dau n mulțimi. Să se genereze în ordine lexicografică elementele produsului cartezian al mulțimilor date.
21. Fie n un număr natural nenul și mulțimea $A = \{1, 2, 3, \dots, n\}$. Să se determine toate partițiile disjuncte ale mulțimii A .
22. Se dă un număr natural n . Determinați, în ordine lexicografică, toate modalitățile de a-l scrie pe n ca sumă de numere naturale.
23. Se dă un număr natural n . Determinați, în ordine lexicografică, toate modalitățile de a-l scrie pe n ca sumă de numere naturale distincte.
24. Se dă un număr natural n și un număr m . Determinați, în ordine lexicografică, toate modalitățile de a-l scrie pe n ca sumă de cel puțin m numere naturale distincte.
25. Să se genereze toate funcțiile surjective definite pe A cu valori în B , unde $\text{card}A = n$ și $\text{card}B = m$ ($n > m$).
26. Să se genereze toate funcțiile injective definite pe A cu valori în B , unde $\text{card}A = n$ și $\text{card}B = m$ ($n < m$).
27. Determinați toate funcțiile surjective $f: \{1, 2, \dots, n\} \rightarrow \{-1, 0, 1\}$, astfel încât $f^2(1) + f^2(2) + \dots + f^2(n) = m$, unde m este dat.
28. Se dă un număr natural n . Determinați, în ordine lexicografică, toate modalitățile de a-l scrie pe n ca sumă de numere naturale ordonate crescător, astfel încât diferența dintre doi termeni consecutivi ai sumei să fie cel puțin 2.
29. Se dă un număr natural n și o mulțime cu m elemente, numere naturale nenule. Determinați, în ordine lexicografică, toate modalitățile de a-l scrie pe n ca sumă de termeni din acea mulțime.
30. Să se descompună un număr natural n în sumă de numere naturale prime între ele.
31. Se dă un număr natural n și un interval $[a, b]$. Determinați, în ordine lexicografică, toate modalitățile de a-l scrie pe n ca sumă de numere naturale din intervalul $[a, b]$.

32. Să se afișeze toate numerele formate din cifre distincte cu proprietatea că suma cifrelor este s . Valoarea variabilei s se citește de la tastatură. Soluțiile se vor afișa într-un fișier text.
33. Să se afișeze toate modurile posibile de a descompune un număr natural n în sumă de p numere naturale distincte (n și p sunt date). Soluțiile se vor scrie în fișierul "*Descompunere.txt*", numerele care alcătuiesc fiecare descompunere pe câte un rând, separate prin spații.
34. Să se descompună un număr n în sumă de numere prime distincte.
35. Să se afișeze toate numerele de n cifre ($n < 10$) pentru care suma cifrelor este egală cu $3n$.
36. Se citește de la tastatură un număr natural n . Să se genereze toate combinațiile de câte n cifre binare care nu au două cifre de 1 alăturate. Rezultatele vor fi trecute în fișierul "*Binar.txt*".
37. Să se genereze numerele de n cifre cu proprietatea că fiecare cifră diferă de precedenta printr-o unitate.
38. Generați toate șirurile de n paranteze rotunde care se închid corect.
39. Se citește de la tastatură un număr natural n ($n < 10$) și apoi n valori naturale $a_1, a_2, \dots, a_n$. Afișați într-un fișier text toate posibilitățile de a intercala între toate numerele $a_1, a_2, \dots, a_n$ operatorii $+$ și $-$ astfel încât, evaluând expresia obținută de la stânga la dreapta, la fiecare pas, rezultatul obținut să fie strict pozitiv. Fiecare soluție se va afișa pe câte o linie a fișierului.
40. Un călător trebuie să se aprovizioneze de la un magazin pentru călătoria pe care urmează să o facă. Produsele existente în magazin, pe care intenționează să le cumpere sunt numerotate cu $1, 2, \dots, n$. Pentru fiecare produs se cunoaște tipul, reprezentat printr-un caracter. Determinați toate variantele pe care le are cumpărătorul de a alege produsele pe care le va cumpăra, știind că poate cumpăra maxim x produse de același tip.
41. Un călător întâlnește în călătoria lui un obstacol, fiind nevoit să renunțe la o parte din obiectele pe care le are în sac. Va putea transporta cel mult o greutate care nu depășește limita v . Numărul de obiecte nu poate fi mai mic decât o valoare a , necesară pentru supraviețuire, dar nu poate fi mai mare decât b . Ce posibilități are călătorul? (Obiectele sunt numerotate cu $1, 2, \dots, n$ cu n dat)
42. Într-un grup sunt n elevi, băieți și fete, numerotați $1, 2, \dots, n$. Fiecare elev cunoaște o parte din ceilalți elevi. O relație de cunoștință nu este neapărat reciprocă (dacă elevul x îl cunoaște pe elevul y , asta nu înseamnă că y îl cunoaște pe x). Relațiile de cunoștințe între elevi se dau în fișierul "*Cunoștințe.in*", care conține pe primul rând numărul de elevi n , iar pe fiecare dintre următoarele rânduri, câte o pereche de numere întregi (x, y) , separate prin spații, cu semnificația " x îl cunoaște pe y ".

- Unul dintre elevi are o carte foarte interesantă, pe care toți membrii grupului vor să o aibă pentru a o citi. Cartea circulă printre membrii grupului în felul următor: fiecare elev după ce a primit-o de la altcineva o dă mai departe, dar numai unui elev pe care îl cunoaște. Determinați toate modalitățile prin care cartea să circule pe la fiecare elev exact o singură dată, fiecare elev transmitând-o mai departe numai către o cunoștință, iar în final cartea să ajungă din nou la proprietar. Soluțiile se vor tipări în fișierul "*Cunoștințe.out*", elementele care alcătuiesc fiecare soluție urmând să se scrie pe un rând, separate prin spații.
43. Să se genereze toate modalitățile de plată a unei sume s de bani, având la dispoziție oricâte monezi din fiecare valoare. Numărul tipurilor de monezi, precum și suma s se citesc de la tastatură.
 44. Se dau n bile albe și albastre. Fiecare bilă are o etichetă de tip caracter. Să se afișeze toate posibilitățile de a selecta p bile din care cel puțin a să fie albe.
 45. Să se determine soluțiile ecuației $2x + y + 3z = 40$, unde x, y, z sunt naturale.
 46. Se consideră o listă care conține n întrebări, numerotate $1, 2, \dots, n$ fiecare întrebare fiind caracterizată printr-un punctaj. Să se determine chestionarele care se pot alcătui astfel încât să aibă între a și b întrebări, iar întrebările care alcătuiesc fiecare chestionar să totalizeze minim c și maxim d puncte. Datele de intrare: n, a, b, c, d și punctajele se citesc dintr-un fișier text.
 47. Se dau 6 culori, codificate prin cifrele $1, 2, \dots, 6$. Să se afișeze toate posibilitățile de alcătuire a unor triunghiuri ale căror laturi să fie colorate cu culori dintre cele date, astfel încât: culoarea din mijloc să aibă număr de ordine par, prima culoare să fie x , unde x este un număr întreg cuprins între 4 și 6. Culoarele cu care se colorează laturile triunghiurilor trebuie să fie distincte.
 48. Să se genereze toate anagramele unui cuvânt citit.
 49. Se consideră un cuvânt de maxim 20 de litere. Să se afișeze toate cuvintele ce conțin literele distincte ale cuvântului inițial și nu conțin două vocale sau două consoane alăturate.
 50. Se dau p litere mici ale alfabetului englez. Se cere să se genereze toate cuvintele de lungime n ($n < p + 1$) care nu conțin trei vocale sau trei consoane alăturate. O literă nu se poate repeta în cadrul cuvântului.
 51. Să se genereze toate șirurile de lungime n , formate numai din literele 'B' și 'C', șiruri care să nu aibă două litere 'B' alăturate. Valoarea lui n se citește de la tastatură ($n < 10$). Fiecare șir se va afișa pe un rând al ecranului, fără spații între litere.
 52. Se citesc din fișierul "*Cuvinte.txt*" n cuvinte, având toate aceeași lungime de m caractere și două cuvinte c_1 și c_2 , ambele de lungime p . Să se afișeze toate modalitățile de alegere a k cuvinte din cele n date astfel încât, scriind cuvintele unul sub altul să se poată forma pe prima coloană cuvântul c_1 , iar pe ultima coloană cuvântul c_2 . Dacă există mai multe soluții se vor afișa toate.
 53. Fișierul text "*Cuvinte.in*" conține o frază scrisă pe un singur rând, alcătuită din cuvinte formate din litere mari și separate prin spațiu. Cu cuvintele frazei se pot alcătui propoziții. Să se realizeze un program care generează toate propozițiile care îndeplinesc următoarele condiții: două cuvinte succesive din șir nu pot începe cu aceeași literă și cuvintele de rang par din șir trebuie să înceapă cu o vocală. Soluțiile se vor tipări în fișierul "*Cuvinte.out*". Fiecare propoziție se va scrie pe un rând al fișierului, cuvintele fiind separate prin spațiu.
 54. Scrieți un program care afișează pe ecran toate secvențele de n litere (n număr natural par dat) din mulțimea $\{A, B, C, D\}$ secvențe care se pot construi respectând următoarele reguli: nu se pot așeza două litere identice una lângă alta și litera D trebuie să apară de $n/2$ ori.
 55. Se citesc din fișierul "*Fazan.txt*" numerele naturale n și m ($m \leq n$) și apoi n cuvinte distincte cu cel mult 10 litere fiecare. Să se afișeze în fișierul "*Rezultat.txt*" toate secvențele de câte m cuvinte dintre cele citite care respectă regulile jocului *Fazan* (ultimele 2 litere ale unui cuvânt coincid cu primele 2 litere ale cuvântului următor).
 56. Se citește un cuvânt cu cel mult 10 caractere litere mici distincte și un număr natural n ($n < 10$). Să se genereze și să se afișeze toate cuvintele care se pot obține din cuvânt prin eliminarea a exact n litere.
 57. Se citește un cuvânt format din maxim 6 litere mici distincte. Să se afișeze în ordine lexicografică toate anagramele cuvântului citit care au proprietatea că nu conțin două vocale alăturate și nici două consoane alăturate. Dacă acest lucru nu este posibil, se va afișa un mesaj corespunzător.
 58. La un concurs național de dans s-au înscris n concurenți, cu numerele de concurs $1, 2, \dots, n$. Pentru fiecare dansator se cunoaște localitatea de origine. În fiecare zi a competiției, vor intra în concurs x concurenți. Afișați toate posibilitățile de a stabili ordinea intrării în concurs a celor x concurenți, știind că: doi dansatori din aceeași localitate nu pot evolua unul după altul și se va respecta ordinea descrescătoare a numerelor cu care intră în concurs. Rezultatele se vor afișa în fișierul "*Dansatori.txt*".
 59. Se dau un număr natural n ($n < 7$) și $2n$ numere naturale cu cel mult 3 cifre fiecare, reprezentând înălțimile în centimetri a $2n$ elevi. Să se aranjeze elevii pe două rânduri astfel încât fiecare elev, în afară de primul de pe rând să aibă în dreapta lui un elev mai înalt decât el și fiecare elev de pe primul rând să aibă în spatele lui unul mai înalt decât el.
 60. Să se afișeze toate modalitățile de așezare a $2n$ persoane, numerotate $1, 2, \dots, n$ astfel încât persoanele din cele două linii să fie așezate în ordine crescătoare și fiecare persoană din prima linie să fie mai mică decât persoana corespunzătoare din linia a doua.
 61. Se dă o matrice pătratică cu n linii și n coloane cu elemente numere naturale distincte. Determinați cea mai mare sumă a n elemente din matrice, cu proprietatea că oricare două elemente se află pe linii și coloane distincte.

62. Fiind dată o mulțime de n pătrate, pentru fiecare pătrat se cunoaște lungimea laturii sale și culoarea. Să se genereze toate dreptunghiurile care se pot forma cu x pătrate, astfel încât 2 pătrate vecine să nu aibă aceeași culoare.
63. Se consideră o tablă de șah de dimensiune $n \times n$. Să se plaseze pe tablă n regine astfel încât să nu existe două regine care să se atace.
64. Să se afișeze toate modurile în care se poate perfora un bilet de autobuz. Un bilet are 9 puncte de perforare posibile.
65. Se citește de la tastatură un număr natural n . Să se genereze toate matricele pătratice binare de dimensiuni $n \times n$, cu proprietatea că în fiecare coloană există o cifră 0. Matricele-soluție se vor scrie în fișierul text "Matrice.out" astfel: elementele fiecărei linii a unei matrice pe câte un rând separate prin spații, iar între două matrice se va lăsa câte un rând liber.
66. Pe o tablă de șah cu $n \times n$ pătrate, poziția pieselor este marcată cu valoarea 1, iar pozițiile libere prin valoarea 0. Pe o poziție x de pe prima linie se află un pion. Să se determine toate traseele pe care poate ajunge pionul până la ultima linie. Se cunoaște posibilitatea de înaintare a pionului (o poziție înainte pe linie sau pe diagonală). Traseele vor fi afișate ca matrice.
67. Pe n scaune sunt aranjate n persoane într-o ordine bine precizată. Să se rearanjeze cele n persoane pe cele n scaune astfel încât două persoane alăturate să nu fi fost alăturate în aranjarea inițială și fiecare persoană trebuie să stea pe un scaun diferit față de poziția inițială.

6.3. Backtracking în plan

1. Pe o tablă de șah de dimensiune $n \times n$ se află un cal în poziția (x_0, y_0) . Să se afișeze toate posibilitățile de a parcurge tabla de șah, fără a trece de două ori prin aceeași poziție.
2. Pe o tablă de șah de dimensiune $n \times n$ se află un cal în poziția (x_0, y_0) . Să se afișeze toate posibilitățile calului de a ajunge în poziția (x_1, y_1) , fără a trece de două ori prin aceeași poziție.
3. Pe o tablă de formă dreptunghiulară se află un cal. Se cunosc coordonatele inițiale ale calului (x_i, y_i) . Acesta trebuie să ajungă într-o poziție finală (x_f, y_f) . Pe tabla de șah se găsesc și alte piese care trebuie ocolite. Coordonatele unde se găsesc aceste piese sunt afișate într-un fișier text. Să se determine câte posibilități sunt ca să ajungă în poziția finală.
4. Fiind dată o tablă de șah de dimensiunea $n \times n$ și un cal în colțul stânga sus al acesteia, se cere să se afișeze toate posibilitățile de mutare a acestei piese de șah, astfel încât să treacă o singură dată prin fiecare pătrat al tablei. O soluție va fi afișată ca o matrice $n \times n$ în care sunt numerotate săriturile calului. Soluțiile problemei vor fi afișate într-un fișier text.

5. Un șoricel se găsește într-un labirint de formă dreptunghiulară cu m linii și n coloane. Labirintul este reprezentat printr-o matrice în care pereții sunt marcați cu 1 și culoarele cu 0. Se cunosc coordonatele inițiale ale șoricelului (x_0, y_0) . Să se determine toate posibilitățile pe care le are șoricelul pentru a ieși din labirint. Șoricelul poate avansa pe 4 direcții, câte o celulă (sus, jos, stânga, dreapta).
6. Un șoricel se găsește într-un labirint de formă dreptunghiulară cu m linii și n coloane. Labirintul este reprezentat printr-o matrice în care pereții sunt marcați cu 1 și culoarele cu 0. Se cunosc coordonatele inițiale ale șoricelului (x_0, y_0) . Să se determine toate posibilitățile pe care le are șoricelul pentru a ajunge la bucata de cașcaval. Coordonatele bucății de cașcaval sunt (x_1, y_1) .
7. Doi prieteni se găsesc într-un labirint, reprezentat printr-o matrice cu $n \times m$ elemente care pot fi 0 sau 1 (culoarele sunt reprezentate prin 0 și pereții prin 1, coordonatele celor doi prieteni sunt (x_1, y_1) și (x_2, y_2)). Să se verifice dacă există posibilitatea ca cei doi prieteni să se întâlnească. În cazul în care există mai multe soluții se vor afișa toate.
8. Fiind dată o matrice cu n linii și n coloane, să se completeze cu numere naturale nenule, astfel încât suma pe orice linie sau coloană, precum și pe diagonalele principală și secundară să fie aceeași. Nu există valori egale pe o linie sau pe o coloană.
9. O fotografie recepționată de un satelit înfățișează o imagine. Pe un anumit fundal se disting unul sau mai multe obiecte. Fotografia este prezentată sub forma unei matrice binare, în care punctele corespunzătoare obiectelor se reprezintă prin valoarea 1, iar cele care aparțin fundalului sunt identificate prin valoarea 0. Două puncte din fotografie aparțin aceluiași obiect, dacă ambele au valoarea 1 și sunt vecine pe linie, coloană sau pe una din diagonale. Să se afișeze toate obiectele din fotografie, apoi să se coloreze aceste obiecte cu culori diferite, simbolizate prin numere $2, 3, 4, \dots, n + 1$ (n este numărul de obiecte). (Problema fotografiei)

6.4. Metoda Greedy

1. Fiind dat un șir de n numere întregi, să se determine cele mai mari două elemente ale șirului.
2. Fie un șir de numere întregi, ale cărui elemente se citesc pe rând de la tastatură până la introducerea valorii 0 (aceasta nu face parte din șir). Să se determine cele mai mici două elemente ale șirului.
3. Se dă un șir de n numere întregi care conține cel puțin un număr strict pozitiv. Să se determine un subșir de sumă maximă având un număr minim de elemente.
4. Se dă un șir de n numere naturale. Să se determine numărul minim de subșiruri strict crescătoare în care se poate partiționa șirul.

5. Se dă un șir cu n numere întregi. Determinați cel mai mare număr care poate fi scris ca produs de două elemente ale șirului.
6. Se dă un șir cu n elemente, numere întregi și un număr natural k ($k \leq n$). Calculați cea mai mare sumă care poate fi obținută schimbând semnul a exact k elemente aflate pe poziții distincte din șirul dat.
7. Se citește de la tastatură un șir de n numere naturale. Să se determine un grup maxim de elemente ale șirului cu proprietatea că elementele sunt numere prime, iar suma lor este egală cu cel mult k (k este un număr dat).
8. Se dau n numere naturale. Să se găsească o subsecvență, astfel încât suma elementelor din această subsecvență să fie divizibilă cu n .
9. Se citește de la tastatură un șir de n numere întregi. Să se descompună acest șir în număr minim de subșiruri strict crescătoare, astfel încât numerele lor de ordine din șirul dat să fie ordonate crescător în subșirurile date.
10. Se citește de la tastatură un șir de n numere naturale. Să se determine un subșir al șirului dat cu proprietatea că toate elementele subșirului sunt numere prime și că suma elementelor subșirului este cel mult egală cu o valoare x dată.
11. Se consideră un număr natural n . Să se scrie n ca o sumă cu un număr minim de termeni diferiți din șirul lui *Fibonacci*.
12. Numerele *frumoase* sunt acele numere naturale care au ca factori primi doar pe 2, 3 și 5. Fiind dat un număr natural n , se cere să se afișeze primele n numere *frumoase*.
13. O persoană are la dispoziție un rucsac cu ajutorul căruia poate transporta o greutate maximă G și urmează să efectueze un transport în urma căruia trebuie să obțină un câștig. Persoana are la dispoziție n obiecte. Pentru fiecare obiect se cunoaște greutatea (mai mică decât capacitatea rucsacului) și câștigul obținut în urma transportului său. Stabiliți ce obiecte trebuie să aleagă persoana, astfel încât câștigul obținut să fie maxim și care este acest câștig.
14. Se consideră un număr nelimitat de rucsaci de același tip, cu fiecare putându-se transporta o greutate maximă G și n ($n \leq 1000$) obiecte de greutate $g_1, g_2, \dots, g_n$. Să se determine numărul minim de rucsaci necesari pentru a transporta toate obiectele.
15. Într-un magazin sunt n obiecte, pentru fiecare se cunoaște greutatea G și valoarea V . O persoană intră în magazin având un rucsac ce poate transporta o greutate maximă G_{Max} . El va lua anumite obiecte, sau porțiuni de obiecte, astfel încât suma greutăților obiectelor luate să nu depășească G_{Max} .
Să se stabilească câștigul maxim pe care îl poate obține persoana. Câștigul este egal cu suma valorilor obiectelor luate. Câștigul adus de o fracțiune de obiect este direct proporțional cu greutatea fracțiunii.
16. La un festival sunt programate n spectacole, pentru fiecare se cunoaște momentul de început și momentul de sfârșit, exprimate prin numere naturale. Un spectator dorește să urmărească cât mai multe spectacole în întregime. Determinați numărul maxim de spectacole care pot fi urmărite, fără ca acestea să se suprapună.
17. Un cabinet medical trebuie să satisfacă cererile a n pacienți. Se cunoaște timpul alocat fiecărui pacient. Să se realizeze un program care stabilește o ordine de intrare a pacienților, astfel încât timpul total de așteptare să fie minim.
18. Fie n și k două numere naturale și S o sumă de bani ce trebuie plătită folosind monezi de valori $k_1, k_2, \dots, k_n$. Presupunând că din fiecare tip de monedă există o cantitate nelimitată, să se determine o modalitate de plată a sumei S , folosind un număr minim din monezi.
19. Un tâmplar vrea să construiască o piesă cu o formă mai ciudată. Pentru aceasta folosește n plăci de lemn de dimensiuni diferite, descrise ca niște intervale închise cu capetele numere reale. Să se găsească o mulțime minimă de cuie astfel încât fiecare placă să fie bătută de cel puțin un cui. Se cere poziția cuielor.
20. Se dau n șiruri de numere întregi ordonate crescător, de dimensiuni $d_1, d_2, \dots, d_n$. Dacă se interclasează șirurile de dimensiuni d_i, d_j atunci se obține un șir de dimensiune $d_i + d_j$. Să se interclasezeze cele n șiruri folosind $n - 1$ pași, la fiecare pas interclasându-se câte două șiruri. Să se obțină un singur șir ordonat, efectuând un număr minim de operații. Să se determine numărul minim de operații necesare pentru a interclasa cele n șiruri.
21. Se citesc de la tastatură două mulțimi de numere întregi $A = \{a_1, a_2, \dots, a_m\}$ și $B = \{b_1, b_2, \dots, b_n\}$, cu m respectiv n elemente. Determinați o submulțime X a lui B cu m elemente de forma $X = \{x_1, x_2, \dots, x_m\}$, astfel încât valoarea expresiei $a_1 \cdot x_1 + a_2 \cdot x_2 + \dots + a_m \cdot x_m$ să fie maximă.
22. Se dau două șiruri de lungime n și un număr k . Cele două șiruri au doar elementele 1 și -1 . Să se transforme primul șir în al doilea, dacă se poate. Singura operație permisă este de a selecta o secvență de k elemente alăturate inversându-le semnul la toate numerele cuprinse în această secvență. Să se afișeze numărul minim de operații și poziția de început a fiecărei operații efectuate.
23. Se dau n șiruri $S_1, S_2, \dots, S_n$ ($n > 1$) de lungimi $L_1, L_2, \dots, L_n$, ordonate crescător. Se cere să se interclaseze șirurile, obținându-se un nou șir, astfel încât numărul de comparații între elementele șirurilor date să fie minim.
24. Se consideră un număr natural nenul n și o funcție $f: A \rightarrow A, A = \{1, 2, \dots, n\}$. Să se determine o submulțime S a lui A cu număr maxim de elemente astfel încât restricția lui f la S este o funcție injectivă $f(S) = S$.

25. Se citește de la tastatură o matrice pătratică a cu n linii și n coloane. Să se scrie un program care, efectuând o succesiune de operații de interschimbare a două linii între ele, sau a două coloane, rearanjează liniile și coloanele matricei, astfel încât suma elementelor de pe diagonala principală să fie minimă.
26. Fiind dată o hartă cu n țări, să se găsească o metodă de colorare a hărții, folosind cel mult patru culori, astfel încât două țări cu frontieră comună să fie colorate diferit (*Problema colorării hărților*).
27. Se consideră o mulțime Q cu n puncte în plan (date prin coordonatele lor carteziane). Se cere determinarea înfășurătorii lor convexe, adică cel mai mic poligon convex P pentru care fiecare punct dat este fie pe conturul lui P , fie în interiorul acestuia. (*Înfășurătoarea convexă a n puncte*)
28. Se consideră n orașe numerotate $1, 2, \dots, n$. Un comis voiajor trebuie să-și prezinte produsele în toate cele n orașe, plecând dintr-un oraș de start, trecând prin fiecare oraș exact o singură dată și revenind în orașul din care a plecat. Știind că între unele dintre orașe există drumuri directe, iar între altele nu, să se afișeze un traseu de lungime minimă pe care îl poate urma comis-voiajorul. Drumurile directe dintre orașe sunt memorate într-o matrice a cu n linii și n coloane, în care fiecare element de pe linia i și coloana j este 1, dacă există drum direct între orașele i și j , respectiv 0 în caz contrar. (*Problema comis-voiajorului*)
29. Se dă o tablă de șah de $n \times n$ pătrate. Să se determine punctele prin care trece un cal, care pornește din punctul $(1, 1)$ și încearcă să acopere cât mai multe puncte posibile.
30. La inaugurarea unui restaurant sunt prezente mai multe persoane. Clienții își aleg din meniul pus la dispoziție câte o specialitate. Restaurantul a angajat un singur bucătar care pregătește mâncărurile una după alta, deci clienții nu pot fi serviți decât pe rând. Presupunând că bucătarul se apucă de lucru după ce s-au strâns toate comenzile, stabiliți în ce ordine trebuie să pregătească specialitățile, astfel încât timpul mediu de așteptare al clienților să fie minim. Fișierul "*Meniu.txt*" va conține pe prima linie un număr natural n , reprezentând numărul clienților. Următoarea linie va conține n numere întregi, reprezentând timpul necesar pregătirii mâncărurilor comandate, în ordine, pentru cele n persoane. Numerele din fișier sunt separate prin spațiu.
31. O asociație caritabilă asigură consultații medicale gratuite pentru cei fără posibilități materiale. Există un singur cabinet dotat cu aparatură medicală. Din acest motiv, la un moment dat un singur medic poate face consultații. Asociația apelează la n medici de diverse specialități, care își oferă benevol serviciile. Fiecare prezintă un singur interval orar în care poate lucra $[s_i, f_i]$ de-a lungul aceleiași zile. Ajutați asociația să realizeze o planificare a consultațiilor în cabinet, astfel încât numărul de medici să fie maxim. Datele se citesc din fișierul "*Medici.txt*". Pe prima linie se găsește un număr n care reprezintă numărul medicilor. A doua și a treia linie conțin orele de început și de sfârșit corespunzătoare celor n medici. Numerele sunt separate prin câte un spațiu.

32. La un colegiu s-au adus n calculatoare. Ele s-au dispus în diverse încăperi: bibliotecă, secretariat, săli de curs, birouri etc. Se hotărăște legarea calculatoarelor într-o rețea liniară: fiecare calculator va fi legat cu alte două calculatoare, într-un lanț, cu excepția calculatoarelor de la capetele lui, care se vor conecta doar cu câte un singur calculator. Cunoșcându-se coordonatele x și y ale fiecărui calculator, să se afișeze o modalitate de legare a calculatoarelor, astfel încât cablul folosit să aibă lungime minimă. Datele de intrare se citesc din fișierul "*Rețea.txt*". Prima linie va conține un număr n , reprezentând numărul calculatoarelor. Următoarele n linii conțin perechi de numere reale strict pozitive x și y , reprezentând coordonatele în plan ale calculatoarelor.
33. Se consideră n camere distincte situate una după alta, astfel încât din camera i ($i \in \{1, 2, \dots, n-1\}$) se poate trece în camera $i+1$. În fiecare cameră i se află inițial m_i mere și p_i pere, cu $m_i > 0$, $p_i > 0$. În plus, $m_n = p_n = 0$.
O persoană cu un rucsac suficient de încăpător, inițial gol, pornind din prima cameră, trece succesiv prin toate celelalte. La intrarea în fiecare cameră descarcă rucsacul, se odihnește și apoi trebuie să încarce fie toate merele, fie toate perele din camera respectivă, după care trece în camera următoare. Să se determine tipul fructului încărcat în fiecare cameră, astfel încât, în ultima cameră să descarce un număr maxim posibil de fructe. (*Problema mere-pere*)

6.5. Programarea dinamică

1. Se dau două numere n și S . Să se determine câte numere de n cifre au suma cifrelor S .
2. Se citește de la tastatură un număr natural n , apoi un șir de n numere întregi. Să se găsească cel mai lung subșir al șirului citit, care are proprietatea că elementele sunt în ordine crescătoare.
3. Se citește de la tastatură un număr natural n și apoi n numere întregi. Să se determine secvența din vector care are suma elementelor maximă.
4. Să se găsească cea mai lungă secvență comună pentru două șiruri.
5. Să se găsească cea mai lungă secvență ordonată, comună pentru două șiruri.
6. Se dă un șir $S = (s_1, s_2, \dots, s_n)$ de lungime n ($1 \leq n \leq 1000$). Un subșir al șirului este de forma: $S' = (s_{i_1}, s_{i_2}, \dots, s_{i_k})$, $i_1 < i_2 < \dots < i_k$. Se cere să se determine un subșir al șirului S , care este ordonat strict crescător și care are lungimea maximă.
7. Din fișierul "*Valori.in*" se citește de pe primul rând un număr natural n , de pe al doilea rând se citesc n numere naturale. Să se calculeze și să se afișeze în fișierul "*Valori.out*" numărul de subșiruri strict crescătoare care se pot forma din șirul citit.

8. Să se calculeze câte submulțimi cu k elemente are o mulțime cu n elemente numere întregi.
9. Un om dorește să urce o scară cu n trepte ($1 < n < 100$). El poate urca una sau două trepte la un moment dat. În câte moduri poate urca această scară? Programul va citi valoarea n de la tastatură și va afișa pe ecran rezultatul cerut.
10. Se consideră un triunghi de numere care are pe prima linie un număr, pe a doua linie două numere, pe a treia linie trei numere, triunghiul având în total n linii ($1 \leq n \leq 1000$). Scrieți un program care să calculeze cea mai mare dintre sumele numerelor ce apar pe drumurile ce pleacă din vârș și ajung la bază, astfel încât în fiecare drum succesorul unui număr se află pe rândul de mai jos dedesubt sau pe diagonală la dreapta. Datele de intrare se citesc din fișierul "*Triunghi.in*", începând cu numărul de linii și continuând cu triunghiul de numere. Programul va afișa pe ecran valoarea cerută.
11. Se dă un șir de n ($1 < n < 1000$) numere naturale, mai mici sau egale cu 1000. Să se împartă elementele din șir în două submulțimi disjuncte, astfel încât sumele elementelor din cele două submulțimi să fie cât mai apropiate.
12. Se consideră un triunghi de numere naturale format din n linii. Prima linie conține un număr, a doua linie conține 2 numere etc. ultima linie n , conține n numere. În acest triunghi se pot calcula diverse sume cu n elemente, astfel:
 - termenul i al sumei se află pe linia i din triunghi;
 - pentru un anumit termen al sumei, termenul următor se află pe linia următoare și pe aceeași coloană, sau pe coloana imediat următoare spre dreapta.
 Să se determine cea mai mică sumă care se poate obține în acest mod și numerele care o alcătuiesc.
13. Se citește dintr-un fișier "*Matrice.in*" un număr natural n , apoi o matrice cu $n \times n$ elemente numere întregi de maxim 4 cifre. Parcurgând matricea din colțul $(n, 1)$ spre colțul $(1, n)$ pe direcțiile nord, nord-est și est, să se afișeze numărul de moduri în care se poate ajunge din colțul $(n, 1)$ în colțul $(1, n)$. Să se afișeze suma maximă care se poate obține parcurgând matricea din colțul $(n, 1)$ în colțul $(1, n)$.
Rezultatul va fi afișat în fișierul "*Matrice.out*".
14. Din fișierul "*Matrice.in*" se citesc două numere naturale n și m mai mici decât 100, apoi o matrice cu $n \times m$ elemente numere întregi. Să se afișeze pentru fiecare coloană a matricei, numărul de elemente al celui mai lung subșir crescător ce se poate forma parcurgând elementele coloanei de jos în sus. Rezultatul va fi trecut în fișierul "*Matrice.out*".
15. Se dă o matrice pătratică cu $n \times n$ elemente numere naturale care are liniile și coloanele numerotate de la 1 la n . Se citește un număr natural m și n perechi de poziții din matrice de forma (i_1, j_1) și (i_2, j_2) astfel încât $i_1 < i_2$ și $j_1 < j_2$. Să se calculeze și să se afișeze pentru fiecare pereche de poziții suma maximă a elementelor din matrice aflate în submatricea care are colțul stânga-sus în (i_1, j_1) și colțul dreapta-jos în (i_2, j_2) .
16. Se dă o expresie booleană exprimată prin șirurile de caractere: "*true*", "*false*", "*and*", "*or*". Să se identifice numărul de parantezări complete (moduri în care se pot așeza parantezele), astfel încât rezultatul final să fie "*true*". Se vor folosi regulile din logica matematică.
17. Se dau două șiruri S_1 și S_2 conținând doar litere mici ale alfabetului englez și având n și, respectiv m caractere ($1 \leq n, m \leq 2000$). Distanța de editare (Levenstein) a celor două șiruri este numărul minim de operații ce trebuie aplicate asupra primului șir pentru a-l transforma în cel de-al doilea șir. Operațiile permise sunt: inserarea unui caracter, ștergerea unui caracter, înlocuirea unui caracter cu un alt caracter. Un tip de operație poate fi aplicat de mai multe ori. Fișierul "*Editare.in*", va conține pe prima linie șirul S_1 , iar pe linia a doua șirul S_2 . Programul va afișa în fișierul "*Editare.out*" un număr întreg reprezentând distanța de editare a celor două șiruri.
18. Se dă o mulțime formată din n ($1 \leq n \leq 1000$) obiecte, fiecare fiind caracterizat de o greutate și un profit. Să se găsească o submulțime de obiecte astfel încât suma profiturilor lor să fie maximă, iar suma greutăților lor să nu depășească o valoare G ($1 \leq G \leq 1000$). Fișierul "*Rucsac.in*", va conține pe prima linie valorile n și G . Pe următoarele n linii se vor găsi perechile de valori W_i și P_i ($0 \leq W_i, P_i \leq 10000$), reprezentând greutatea, respectiv profitul obiectului i . Programul va afișa o singură valoare P_{max} , profitul maxim care poate fi obținut.
19. Se consideră o tablă de șah de dimensiune $n \times n$ ($n \leq 100$) care are așezate piese pe diferite poziții. Se cere să se afișeze numărul minim de mutări necesare unui nebun pentru a ajunge dintr-o poziție inițială dată, într-o poziție finală dată, respectând regulile jocului de șah. Nebunul trebuie să ocolească piesele de pe tabla de șah.
20. Se consideră o frază formată din literele mici ale alfabetului englezesc. Din frază s-au eliminat spațiile dintre cuvinte. Având la dispoziție un dicționar de cuvinte, să se reconstituie fraza, folosind un număr minim de cuvinte (dacă un cuvânt din dicționar apare de mai multe ori, se numără de fiecare dată). Fișierul "*Frazal.txt*" conține pe prima linie numărul n de cuvinte. Pe fiecare dintre următoarele n linii se află câte un cuvânt din dicționar. Fișierul "*FrazaF.txt*" conține pe prima linie numărul m al literelor din frază. Pe a doua linie se află fraza din care au fost eliminate spațiile. Caracterele din frază nu sunt despărțite prin spații. Rezultatele obținute se vor scrie în fișierul "*Fraza.out*".
21. Se consideră un șir de cuvinte formate din litere mici ale alfabetului englez. Să se determine ce mai lung șir de cuvinte din șirul inițial, care are proprietatea că fiecare cuvânt din șir, mai puțin primul, începe cu litera cu care se termină cuvântul anterior și cuvintele din șirul determinat sunt în aceeași ordine în care se află în șirul inițial. Fișierul text "*Sir.in*" conține pe primul rând un număr n , care reprezintă numărul de cuvinte. Pe fiecare din următoarele n rânduri se găsește câte un cuvânt. Rezultatul se va afișa în fișierul "*Sir.out*", astfel încât pe primul rând se va găsi numărul de cuvinte din care este constituit cel mai lung șir, iar pe următoarea linie cuvintele folosite, despărțite prin spațiu.

6.6. Probleme olimpiadă

Problema 1 - Summax

Avem o matrice triunghiulară cu n linii, cu elemente numere întregi. În această matrice putem construi un traseu după următoarea regulă:

- primul element al traseului este elementul $a_{1,1}$
- dacă elementul $a_{i,j}$ aparține traseului, atunci următorul element al traseului poate fi doar $a_{i+1,j}$ sau $a_{i+1,j+1}$, pentru orice $1 \leq j \leq i < n$.

Traseul se va codifica cu numerele de ordine ale coloanelor, parcurgând liniile de la 1 la n . Valoarea traseului este egală cu suma elementelor ce îl formează.

Traseul evidențiat în exemplul din dreapta are valoarea $5+4+6+5+4=24$, și se codifică cu 1,2,3,3,4.

Fie mulțimea tuturor traseelor de valoare maximă generate în ordine lexicografică și numerotate. Pentru exemplul alăturat avem șase trasee de lungime maximă:

a_{11}				
a_{21}	a_{22}			
a_{31}	a_{32}	a_{33}		
a_{41}	a_{42}	a_{43}	a_{44}	
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

5				
2	4			
7	5	6		
6	6	5	5	
3	4	3	4	4

traseul 1.	1 1 1 1 2	(5+2+7+6+4=24)
traseul 2.	1 1 1 2 2	(5+2+7+6+4=24)
traseul 3.	1 2 2 2 2	(5+4+5+6+4=24)
traseul 4.	1 2 3 3 4	(5+4+6+5+4=24)
traseul 5.	1 2 3 4 4	(5+4+6+5+4=24)
traseul 6.	1 2 3 4 5	(5+4+6+5+4=24)

Cerință

Cunoscând dimensiunea și elementele unei matrice triunghiulare, respectiv două numere naturale st și dr ($st \leq dr$), se cere să se determine:

1. Numărul total al traseelor de valoare maximă. În cazul în care această valoare depășește 2000000000, se va tipări valoarea 2000000001;
2. Traseele cu numerele de ordine $st, st+1, \dots, dr$.

Date de intrare

Fișierul `summax.in` conține pe prima linie un număr natural v . Pentru toate testele de intrare, numărul v poate avea doar valoarea 1 sau 2.

A doua linie conține trei numere naturale n, st și dr , separate prin spațiu. Următoarele n linii conțin câte o linie a matricei triunghiulare astfel: linia i conține i elemente, și anume valorile $a_{i,1} a_{i,2} \dots a_{i,i}$ pentru orice $1 \leq i \leq n$.

Date de ieșire

Dacă valoarea lui v este 1, se va rezolva numai punctul 1 din cerință.

În acest caz, în fișierul de ieșire `summax.out` se va scrie un singur număr natural ce reprezintă numărul traseelor de lungime maximă.

Dacă valoarea lui v este 2, se va rezolva numai punctul 2 din cerință.

În acest caz, în fișierul de ieșire `summax.out` se vor tipări pe câte o linie n numere naturale separate prin spațiu, reprezentând codificările traseelor de valoare maximă cu numerele de ordine $st, st+1, \dots, dr$.

Restricții și precizări

- $1 \leq n \leq 2000$
- $1 \leq st \leq dr \leq 2\,000\,000\,000$
- $1 \leq dr - st \leq 1000$
- elementele matricei triunghiulare sunt numere naturale strict pozitive
- valoarea maximă a traseului nu depășește 1 000 000 000

Exemple

summax.in	summax.out	explicație
1	6	$v=1$
5 2 4		Numărul traseelor de valoare maximă este 6.
5		
2 4		(vezi exemplul de mai sus).
7 5 6		
6 6 5 5		
3 4 3 4 4		

summax.in	summax.out	explicație
2	1 1 1 2 2	$v=2$
5 2 4	1 2 2 2 2	$st=2$ $dr=4$
5	1 2 3 3 4	S-au tipărit traseele cu numerele de ordine 2, 3 și 4.
2 4		(vezi exemplul de mai sus).
7 5 6		
6 6 5 5		
3 4 3 4 4		

Alte informații utile

- o variabilă de tip `int` (C++) respectiv `integer` (Pascal) ocupă 4 octeți (32 de biți)
- o matrice cu 1000 de linii și 1000 de coloane cu elemente întregi ocupă $(1000 \cdot 1000 \cdot 4) / (1024 \cdot 1024) = 3,81$ MB
- Testele de intrare au următoarea configurație:

fișierul de intrare

	v	n	st	dr
0	1	20	4	10
1	1	900	50	100
2	1	1300	2000	3000
3	1	1700	20000	21000
4	2	20	6	9
5	2	30	20 000 000	20 001 000
6	2	60	40 030 000	40 031 000
7	2	100	139876543	139876 999
8	2	500	137 987000	137 988000
9	2	700	123456789	123457777
10	2	900	100000000	100001000
11	2	1000	1999999999	2000000000
12	2	1010	1000000	1000001
13	2	1020	10123	11111
14	2	1100	1999999999	2000000000
15	2	1200	1000000	1000001
16	2	1300	10123	11111
17	2	1500	1999999999	2000000000
18	2	1600	1000000	1000 999
19	2	1700	10123	11 123

(Olimpiada județeană, 2016)

Problema 2 - Charlie

Charlie a decis să se joace cu literele dintr-un șir de caractere, șir ce conține doar literele mici ale alfabetului englez 'a'...'z'. Jocul constă în a elimina litere din șir după următoarea regulă: fie **L1**, **L2**, **L3** trei litere aflate pe poziții consecutive în șir, atunci litera **L2** poate fi eliminată dacă și numai dacă este strict mai mică lexicografic decât literele **L1** și **L3**.

Pentru a face jocul mai interesant, **Charlie** atașează eliminării literei **L2** un cost egal cu valoarea maximă dintre $\delta(L1)$ și $\delta(L3)$, unde prin $\delta(\text{litera})$ înțelegem numărul de ordine al literei respective în alfabet ($\delta('a')=1$, $\delta('b')=2, \dots, \delta('z')=26$). **Charlie** aplică în mod repetat procedeul de eliminare și calculează suma costurilor eliminărilor efectuate.

Cerințe

Fiind dat un șir de caractere să se determine:

- Lungimea maximă a unei secvențe de litere alternante, adică o secvență pentru care literele aflate pe poziții consecutive sunt de forma: $L_i > L_{i+1} < L_{i+2} > L_{i+3} < L_{i+4} > \dots < L_j$.
- Suma maximă pe care o poate obține **Charlie** aplicând în mod repetat procedeul de eliminare a literelor, precum și șirul obținut în final.

Date de intrare

Fișierul de intrare *charlie.in* conține pe prima linie un număr natural **p**. Pentru toate testele de intrare, numărul **p** poate avea doar valoarea 1 sau valoarea 2. Pe următoarea linie se află un șir de caractere.

Date de ieșire

Dacă valoarea lui **p** este 1, se va rezolva numai punctul a) din cerință.

În acest caz, în fișierul de ieșire *charlie.out* se va scrie un singur număr natural **L** ce reprezintă lungimea maximă a unei secvențe de litere alternante.

Dacă valoarea lui **p** este 2, se va rezolva numai punctul b) din cerință.

În acest caz, fișierul de ieșire *charlie.out* va conține două linii. Pe prima linie se va afla șirul rezultat în urma eliminărilor repetate de litere respectând regula enunțată, iar pe cea de-a doua linie suma maximă obținută.

Restricții și precizări

- $3 \leq$ numărul de litere ale șirului inițial ≤ 100000
- Pentru rezolvarea corectă a primei cerințe se acordă 25 de puncte, iar pentru cerința a doua se acordă 75 de puncte.
- Pentru 30% dintre teste numărul de litere ale șirului ≤ 1000

Exemple

charlie.in	charlie.out	Explicație
1 cadgfacbd a	5	p = 1 Secvențele alternante corect formate sunt: cad , facbd . Lungimea maximă este 5 Atenție! Pentru acest test se rezolvă doar cerința a).

charlie.in	charlie.out	Explicație
2 cbcabadb ac	ccdc 21	p = 2 Șirul inițial: cbcabadb Eliminăm din secvența bad litera a și adăugăm la suma valoarea 4 Șirul rezultat în urma eliminării este: cbcabdbac Eliminăm din secvența bac litera a și adăugăm la suma valoarea 3 Șirul rezultat în urma eliminării este: cbcabdbc Eliminăm din secvența dbc litera b și adăugăm la suma valoarea 4 Șirul rezultat în urma eliminării este: cbcabdc Eliminăm din secvența cab litera a și adăugăm la suma valoarea 3 Șirul rezultat în urma eliminării este: cbcbdc Eliminăm din secvența cbd litera b și adăugăm la suma valoarea 4 Șirul rezultat în urma eliminării este: cbcdc Eliminăm din secvența cbc litera b și adăugăm la suma valoarea 3 Șirul rezultat în urma eliminării este: ccdc Nu mai sunt posibile eliminări. Suma maximă obținută este 21 . Atenție! Pentru acest test se rezolvă doar cerința b).

(Olimpiada județeană, 2015)

Problema 3 – Castel

100 puncte

Arheologii au descoperit pe un platou muntos greu accesibil ruinele unui castel medieval, pe care l-au fotografiat din elicopter, obținând harta digitizată a acestuia. Harta este memorată sub forma unui tablou bidimensional H , compus din $N \times N$ pătrate cu latura egală cu unitatea, având ca elemente numere naturale între 0 și 15, care codifică forma pereților fiecărui pătrat unitar. Dacă scriem numărul natural $H[i][j]$ în baza 2, folosind **exact 4** cifre binare, fiecare bit dă informații despre unul dintre pereții posibil de construit pe fiecare latură a pătratului unitar din poziția (i, j) , astfel:

- dacă bitul de pe poziția 0 are valoarea 1, atunci există perete pe latura vestică (latura din stânga);
- dacă bitul de pe poziția 1 are valoarea 1, atunci există perete pe latura sudică (latura de jos);
- dacă bitul de pe poziția 2 are valoarea 1, atunci există perete pe latura estică (latura din dreapta);
- dacă bitul de pe poziția 3 are valoarea 1, atunci există perete pe latura nordică (latura de sus);
- un bit de valoare 0 indică lipsa peretelui corespunzător acestuia;

Pentru un număr scris în baza 2, numerotarea cifrelor începe cu poziția 0, de la dreapta la stânga.

Castelul este interesant deoarece, pentru realizarea unei mai bune apărări, camerele ce-l compun sunt construite fie independent, fie una în interiorul alteia. Orice camera este construită la o distanță de cel puțin o unitate față de zidul ce împrejmuiește castelul sau față de pereții altor camere.

Folosind harta, arheologii doresc să afle informații privind numărul camerelor și camera de arie maximă. Prin arie a unei camere se înțelege numărul pătratelor unitate cuprinse în interiorul pereților acesteia, fără a socoti ariile camerelor construite în interiorul ei.

Cerința

Cunoscând codificarea hărții castelului, să se determine:

1. numărul total al camerelor din castel
2. aria maximă a unei camere
3. coordonatele colțurilor din stânga-sus, respectiv dreapta-jos a camerei cu aria maximă. Dacă există mai multe camere având aceeași arie maximă, atunci se vor afișa coordonatele camerei având colțul din stânga-sus (**lin1**, **col1**) cu **lin1** minimă, iar la linii egale pe aceea cu **col1** minimă.

Date de intrare

Datele de intrare se citesc din fișierul **castel.in**, care are următoarea structură:

- Pe prima linie se află numărul natural **C**, care poate fi egal cu 1, 2 sau 3, în funcție de cerința ce trebuie rezolvată;

- Pe linia următoare se află numărul natural N , reprezentând dimensiunea hărții;
- Pe următoarele N linii se găsesc câte N numere naturale din intervalul $[0,15]$, separate prin câte un spațiu, reprezentând harta castelului.

Date de ieșire

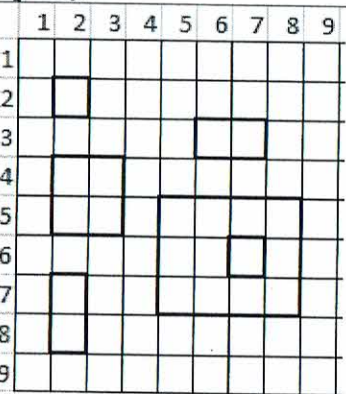
Datele de ieșire se vor scrie în fișierul **castel.out**, astfel:

- Dacă $C = 1$, pe prima linie se va scrie numărul total al camerelor din castel;
- Dacă $C = 2$, pe prima linie se va scrie aria maximă a unei camere din castel;
- Dacă $C = 3$, pe prima linie se vor scrie 4 numere naturale **lin1 col1 lin2 col2**, separate prin câte un spațiu, reprezentând coordonatele colțurilor din stânga-sus, respectiv dreapta-jos ale camerei de arie maximă.

Restricții și precizări:

- $2 < n \leq 100$
- Se garantează că în castel există cel puțin o cameră;
- Testele care au $C = 1$ totalizează 20 de puncte;
- Testele care au $C = 2$ totalizează 50 de puncte;
- Testele care au $C = 3$ totalizează 20 de puncte;
- Se acordă 10 puncte din oficiu.

Exemplu

castel.in	castel.out	Explicație
<pre> 1 9 0 2 0 0 0 0 0 0 0 4 15 1 0 0 2 2 0 0 0 10 2 0 4 11 14 1 0 4 9 12 1 2 10 10 2 0 4 3 6 5 9 8 10 12 1 0 10 8 4 1 4 15 5 1 4 13 1 4 3 2 10 6 1 4 7 1 0 8 8 8 8 0 0 8 0 0 0 0 0 0 0 </pre>	6	 În figură este reprezentată harta castelului codificat în fișierul de intrare. Acesta conține 6 camere.

<pre> 2 9 0 2 0 0 0 0 0 0 0 4 15 1 0 0 2 2 0 0 0 10 2 0 4 11 14 1 0 4 9 12 1 2 10 10 2 0 4 3 6 5 9 8 10 12 1 0 10 8 4 1 4 15 5 1 4 13 1 4 3 2 10 6 1 4 7 1 0 8 8 8 8 0 0 8 0 0 0 0 0 0 0 </pre>	11	Aria maximă a unei camere este 11.
<pre> 3 9 0 2 0 0 0 0 0 0 0 4 15 1 0 0 2 2 0 0 0 10 2 0 4 11 14 1 0 4 9 12 1 2 10 10 2 0 4 3 6 5 9 8 10 12 1 0 10 8 4 1 4 15 5 1 4 13 1 4 3 2 10 6 1 4 7 1 0 8 8 8 8 0 0 8 0 0 0 0 0 0 0 </pre>	5 5 7 8	Camera cu aria maximă are coordonatele (5,5) – (7,8)

(Olimpiada județeană, 2018)

Problema 4 – Frații 2

100 puncte

Numărul 1 poate fi scris în diverse moduri ca sumă de fracții cu numărătorul 1 și numitorul o putere a lui 2. De exemplu:

$$1 = \frac{1}{2} + \frac{1}{2} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} = \frac{1}{8} + \frac{1}{4} + \frac{1}{2} + \frac{1}{8}$$

Două scrieri nu sunt considerate distincte dacă folosesc aceleași fracții scrise în altă ordine. În exemplul de mai sus ultimele două scrieri nu sunt distincte.

Cerință

Pentru N —număr natural nenul să se determine:

- O modalitate de scriere a numărului 1 ca sumă de exact N fracții cu numărătorul 1 și numitorul o putere a lui 2.
- Numărul de scrieri distincte a numărului 1 ca sumă de exact N fracții cu numărătorul 1 și numitorul o putere a lui 2. Deoarece acest număr poate fi foarte mare acest număr trebuie calculat **modulo 100003**.

Date de intrare

Fișierul de intrare *fractii2.in* conține pe prima linie un număr natural p . Pentru toate testele de intrare, numărul p poate avea doar valoarea 1 sau valoarea 2.

Pe a doua linie se găsește un singur număr N natural – reprezentând numărul de fracții.

Date de ieșire

Dacă valoarea lui p este 1, se va rezolva numai punctul a) din cerință. În acest caz, în fișierul de ieșire *fractii2.out* se vor scrie, pe o singură linie, N numere naturale separate prin câte un spațiu reprezentând cei N exponenți ai lui 2 din scrierea solicitată în prima cerință. Astfel, dacă numerele afișate sunt $m_1, m_2, \dots, m_N$ atunci există scrierea $1 = \frac{1}{2^{m_1}} + \frac{1}{2^{m_2}} + \dots + \frac{1}{2^{m_N}}$.

Dacă valoarea lui p este 2, se va rezolva numai punctul b) din cerință. În acest caz, în fișierul de ieșire *fractii2.out* se va scrie un număr natural reprezentând răspunsul la a doua cerință, adică numărul de scrieri distincte a numărului 1 ca sumă de N fracții cu numărătorul 1 și numitorul o putere a lui 2 (modulo 100003).

Restricții

- $2 \leq N \leq 2000$
- Pentru prima cerință se acordă 20% din punctaj.
- Pentru a doua cerință se acordă 80% din punctaj.
- Rezultatul pentru a doua cerință trebuie afișat modulo 100003

Exemple

fractii2.in	fractii2.out	Explicație
1 4	2 2 2 2	$p=1$ $1 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4}$ Răspunsul corespunde celei de-a doua scrieri dar există și alte variante corecte de răspuns. De exemplu, 3 1 2 3 se consideră răspuns corect. Atenție! Pentru acest test se va afișa doar rezultatul la cerința a).

fractii2.in	fractii2.out	Explicație
2 4	2	$p=2$ $1 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4}$ Acestea sunt singurele scrieri distincte. Atenție! Pentru acest test se va afișa doar rezultatul la cerința b).

(Olimpiada județeană, 2014)

Problema 5 – Compresie

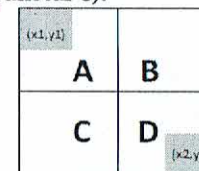
100 puncte

Se consideră un text memorat într-o matrice M , definită prin coordonatele colțului stânga sus $(x1, y1)$ și coordonatele colțului dreapta jos $(x2, y2)$.

Prin aplicarea unui algoritim de compresie, matricei M i se asociază un șir de caractere, notat C_M .

Șirul de caractere C_M este construit prin aplicarea următoarelor reguli:

- a) dacă matricea M are o singură linie și o singură coloană atunci C_M conține numai caracterul memorat în matrice;
- b) dacă toate elementele matricei sunt identice atunci întreaga matrice M se comprimă și C_M este șirul kc , unde k reprezintă numărul de caractere din matrice, iar c caracterul memorat;
- c) dacă matricea este formată din caractere diferite și are cel puțin două linii și două coloane atunci:
 - matricea este împărțită în 4 submatrice A, B, C, D după cum este ilustrat în figura alăturată, unde coordonatele colțului stânga sus ale submatricei A sunt $(x1, y1)$, iar coordonatele colțului dreapta jos sunt $((x2+x1)/2, (y2+y1)/2)$;
 - C_M este șirul $*C_A C_B C_C C_D$ unde C_A, C_B, C_C, C_D sunt șirurile de caractere obținute, în ordine, prin compresia matricelor A, B, C, D utilizând același algoritim;
- d) dacă matricea este formată din caractere diferite, are o singură linie și mai multe coloane atunci C_M este șirul $*C_A C_B$ unde A, B, C_A, C_B au semnificația descrisă la punctul c);
- e) dacă matricea este formată din caractere diferite, are mai multe linii și o singură coloană atunci C_M este șirul $*C_A C_C$ unde A, C, C_A, C_C au semnificația descrisă la punctul c).



Cerință

Dat fiind șirul de caractere C_M ce se obține în urma aplicării algoritmului de compresie asupra unei matrice M de dimensiune $N \times N$ să se determine:

- a) numărul de împărțiri care au fost necesare pentru obținerea textului compresat;
- b) matricea inițială din care provine textul compresat.

Date de intrare

Fișierul de intrare *compresie.in* conține pe prima linie un șir de caractere ce reprezintă textul compresat.

Date de ieșire

Fișierul de ieșire compresie.out conține:

- pe prima linie un număr natural ce reprezintă numărul **nr** de împărțiri care au fost necesare pentru obținerea textului compresat;
- pe următoarele **N** linii se găsesc câte **N** caractere, litere mici ale alfabetului englez, neseparate prin spații, ce reprezintă, în ordine, liniile matricei inițiale.

Restricții și precizări

- $2 \leq N \leq 1000$
- $0 \leq nr \leq 1000000$
- $2 \leq \text{lungimea șirului compresat} \leq 1000000$
- Textul memorat inițial în matricea **M** conține numai caractere din mulțimea literelor mici {a...z}.
- Pentru rezolvarea corectă a cerinței a) se acordă **20%** din punctaj, iar pentru rezolvarea corectă a ambelor cerințe se acordă tot punctajul.

Exemple

compresie.in	compresie.out	Explicații
*4b*bbab4a*abbb	3 bbbb bbab aaab aabb	Au fost efectuate 3 împărțiri: 1) $M = * \begin{pmatrix} b & b \\ b & b \end{pmatrix} \begin{pmatrix} b & b \\ a & b \end{pmatrix} \begin{pmatrix} a & a \\ a & a \end{pmatrix} \begin{pmatrix} a & b \\ b & b \end{pmatrix}$ 2) $\begin{pmatrix} b & b \\ a & b \end{pmatrix} = *(b) \quad 3) \begin{pmatrix} a & b \\ b & b \end{pmatrix} = *(a)(b)(b)(b)$
*4a*ab*aba	3 aaa aab aba	Au fost efectuate 3 împărțiri: 1) $M = * \begin{pmatrix} a & a \\ a & a \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} (a \quad b)(a)$ 2) $\begin{pmatrix} a \\ b \end{pmatrix} = *(a)(b)$ 3) $(a \quad b) = *(a)(b)$

(Olimpiada județeană, 2012)

7. GRAFURI

7.1. Grafuri neorientate

7.1.1 Noțiuni generale

1. Să se memoreze și să se afișeze un graf neorientat folosind cele trei modalități de reprezentare:
 - Matrice de adiacență;
 - Vector de muchii;
 - Liste de vecini.
2. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se scrie un program care afișează gradul unui vârf dat.
3. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se afișeze gradele tuturor vârfurilor.
4. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se afișeze vârfurile de grad par.
5. Se dă lista muchiilor unui graf neorientat. Să se afișeze gradul fiecărui vârf.
6. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se afișeze vârfurile izolate (vârfurile cu gradul 0). Dacă nu are vârfuri izolate, se va afișa un mesaj corespunzător.
7. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se afișeze vârfurile terminale (vârfurile cu gradul 1).
8. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se scrie un program care verifică dacă graful are toate vârfurile interioare (nu sunt nici izolate, nici terminale).
9. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se scrie un program care verifică dacă un vârf dat este terminal, izolat sau interior.
10. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se scrie un program care afișează frecvența vârfurilor terminale, izolate, interioare.
11. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se scrie un program care afișează vecinii unui vârf dat.
12. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se scrie un program care afișează listele vecinilor tuturor vârfurilor.

13. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se scrie un program care afișează vectorul de muchii aferent grafului dat.
14. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin vectorul de muchii. Să se scrie un program care afișează matricea de adiacență pentru graful dat.
15. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin vectorul de muchii. Să se scrie un program care afișează listele vecinilor pentru graful dat.
16. Fie G un graf neorientat, cu n vârfuri. Numim *matricea costurilor* (ponderilor) pentru un graf neorientat, matricea pentru care $a_{ij} = c$ unde c reprezintă costul muchiei (i, j) și 0 în cazul în care nu există muchie (i, j) . Să se reprezinte un graf prin matricea costurilor.
17. Se citește de la tastatură un vector de înregistrări, reprezentând vectorul de muchii al unui graf neorientat G . Fiecare înregistrare va avea trei câmpuri: extremitățile unei muchii și costul asociat ei. Să se construiască matricea costurilor asociată grafului.
18. Se definește o muchie a unui graf neorientat ca o înregistrare cu trei câmpuri: extremitățile unei muchii și un cost asociat muchiei (un număr întreg). Definim un graf neorientat ca un vector de muchii, fiecare element al vectorului având forma descrisă mai sus. Să se afișeze muchia (muchiiile) de *cost minim*.
19. Fie G un graf neorientat cu n vârfuri, reprezentat prin matricea costurilor. Să se afișeze muchia (muchiiile) de *cost minim*.
20. Se definește o muchie a unui graf neorientat ca o înregistrare cu trei câmpuri: extremitățile unei muchii și un cost asociat muchiei (un număr întreg). Definim un graf neorientat ca un vector de muchii, fiecare element al vectorului având forma descrisă mai sus. Să se afișeze *costul mediu* al grafului (media aritmetică a costurilor muchiilor).
21. Fie G un graf neorientat cu n vârfuri, reprezentat prin matricea costurilor. Să se afișeze *costul mediu* al grafului (media aritmetică a costurilor muchiilor).
22. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Fiind dat șirul $d_1, d_2, \dots, d_n$, să se verifice dacă poate reprezenta șirul gradelor vârfurilor, în această ordine.
23. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Fiind dat șirul $d_1, d_2, \dots, d_n$, să se verifice dacă poate reprezenta șirul gradelor vârfurilor, nu neapărat în această ordine.
24. Se citește secvența de numere naturale $d_1 \geq d_2 \geq d_3 \geq \dots \geq d_n$. Să se decidă dacă această *secvență* este *grafică*, adică dacă există un graf neorientat cu n vârfuri ale căror grade să fie chiar numerele $d_1 \geq d_2 \geq d_3 \geq \dots \geq d_n$.
25. Se citește un graf neorientat de la tastatură, precizându-se numărul de vârfuri, numărul de muchii și muchiile. Să se determine gradele tuturor vârfurilor care sunt numere prime.
26. Se citește un graf neorientat de la tastatură, precizându-se numărul de vârfuri, numărul de muchii și muchiile. Să se afișeze pentru un vârf citit, vârfurile adiacente.
27. Se citește un graf neorientat de la tastatură dat prin matricea de adiacență. Să se afișeze vârfurile incidente cu muchia (x, y) din graf.
28. Se citește un graf neorientat de la tastatură dat prin matricea de adiacență. Să se afișeze muchiile incidente cu vârful x din graf.
29. Se consideră un graf neorientat cu n vârfuri și m muchii, dat prin vectorul de muchii. Să se afișeze toate perechile de muchii incidente.
30. Se citește un graf neorientat de la tastatură dat prin listele vecinilor. Să se afișeze vârful (vârfurile) cu cei mai mulți vecini.
31. Se citește dintr-un fișier, de pe primul rând, o valoare n , reprezentând numărul de vârfuri pentru un graf neorientat și de pe următoarele n linii și n coloane matricea de adiacență corespunzătoare grafului.
 - Identificați mulțimea X (*mulțimea vârfurilor*);
 - Identificați mulțimea U (*mulțimea muchiilor*);
 - Calculați gradele vârfurilor impare;
 - Verificați dacă graful are vârfuri izolate.
 În caz afirmativ, se vor afișa aceste vârfuri, în caz contrar se va afișa un mesaj corespunzător.
32. Se dă matricea de adiacență a unui graf neorientat. Să se afișeze, pentru fiecare vârf al grafului, lista vecinilor săi.
33. Se dă lista muchiilor unui graf neorientat. Să se afișeze vârfurile izolate ale grafului.
34. Fișierul "*Graf.in*" conține pe primul rând două numere naturale n și x , reprezentând numărul de vârfuri ale unui graf neorientat, pe fiecare din următoarele n rânduri, câte n valori de 0 și 1 separate prin spații, reprezentând elementele unei linii a matricei de adiacență corespunzătoare grafului. Să se scrie un program care afișează în fișierul "*Graf.out*" vârfurile al căror grad este egal cu valoarea dată x .
35. Se dă lista muchiilor unui graf neorientat. Să se afișeze vârful (vârfurile) de grad maxim.
36. Se consideră matricea de adiacență a unui graf neorientat. Să se afișeze vârful (vârfurile) de grad maxim.
37. Fișierul "*Graf.in*" conține pe prima linie numărul n de vârfuri și numărul de muchii ale unui graf neorientat G , iar pe următoarele n linii, elementele din matricea de adiacență atașată grafului. Să se afișeze în fișierul "*Graf.out*" vârfurile al căror grad se află într-un interval dat $[x, y]$ (x și y sunt numere naturale introduse de la tastatură).

38. Fișierul "Graf.in" conține pe primul rând două valori n și m , reprezentând numărul de vârfuri, respectiv numărul de muchii ale unui graf neorientat, apoi pe fiecare dintre cele m rânduri câte o pereche de numere întregi separate prin spații, reprezentând extremitățile unei muchii. Să se scrie în fișierul "Graf.out" vârfurile grafului, în ordinea descrescătoare a gradelor, apoi să se afișeze câte dintre vârfurile grafului au același grad.
39. Se dă lista muchiilor unui graf neorientat cu n vârfuri, etichetate de la 1 la n . Să se elimine toate muchiile cu proprietatea că ambele extremități au aceeași paritate. Să se afișeze graful astfel obținut.
40. Fiind dată o matrice pătratică de ordin n , să se precizeze dacă poate fi considerată matricea de adiacență a unui graf neorientat cu n vârfuri.
41. Se citește din fișierul "Graf.in" de pe prima linie numărul de vârfuri și numărul de muchii (n, m) , de pe următoarele m rânduri, perechi de vârfuri reprezentând muchiile grafului. Să se construiască matricea de adiacență și să se scrie aceasta în fișierul "Graf.out".
42. Se citește din fișierul "Graf.in" de pe prima linie numărul de vârfuri și numărul de muchii (n, m) , de pe următoarele m rânduri, perechi de vârfuri reprezentând muchiile grafului. Să se calculeze gradul fiecărui vârf și să se memoreze într-un vector. Vectorul va fi afișat în fișierul "Graf.out".
43. Fișierul "Graf.in", conține pe primul rând numărul de vârfuri și numărul de muchii, separate prin spațiu și pe următoarele m rânduri perechile de vârfuri reprezentând muchiile. Formați matricea de adiacență. Enumerați vârfurile terminale și vârfurile intermediare ale grafului. Rezultatul va fi trecut în fișierul "Graf.out".
44. Să se coloreze în toate modurile posibile muchiile unui graf neorientat folosind c culori codificate $1, 2, \dots, c$, dacă acest lucru este posibil. Orice două muchii incidente trebuie să fie colorate diferit.
45. Într-un grup de n persoane se stabilesc perechi de persoane care se consideră prietene. Folosind sintagma că "*prietenul prietenului meu îmi este prieten*", să se determine grupurile care sunt formate dintr-un număr p de persoane între care se pot stabili relații de prietenie, directe sau indirecte (p este citit de la tastatură).
46. Să se realizeze un program care generează matricele de adiacență ale tuturor grafurilor neorientate cu vârfurile $\{1, 2, \dots, n\}$. Rezultatul va fi trecut într-un fișier.
47. Fiind dat un grup de persoane, reprezentate prin numerele $1, 2, \dots, n$ și precizându-se relațiile de simpatie astfel: pentru fiecare persoană i , se citesc numerele de ordine ale persoanelor pe care aceasta le simpatizează (numerele se dau pe aceeași linie, separate prin spații), să se precizeze dacă în grupul de persoane amintit, toate simpatiile sunt reciproce.

48. Vom considera un mod de desfășurare a unui zar, în care fiecare față a zarului este numerotată cu o cifră de la 1 la 6. Desfășurarea zarului se poate realiza în mai multe moduri. Avem, de exemplu, următoarele două moduri de desfășurare:

```

1
5 4 6 2
3

```

```

2
5 1 6 3
4

```

O desfășurare de tipul utilizat în exemplele anterioare se descrie printr-o secvență de șase cifre, citite de sus în jos și, pe fiecare linie, de la stânga la dreapta. Deci, pentru exemplele date, vom avea succesiunea de cifre 154623, respectiv 251634. Scrieți un program care să testeze dacă două perechi de succesiuni de cifre date reprezintă desfășurări diferite ale unui zar.

7.1.2. Grafuri neorientate asociate unui graf dat

- Fiind date matricele de adiacență A și B de dimensiune n , să se verifice dacă matricea B poate reprezenta un subgraf al grafului reprezentat de matricea A .
- Se dau $G = (X, U)$ și $x, y \in X, x \neq y$. Să se determine un subgraf al lui G cu număr minim de vârfuri, care conține pe x și y .
- Fie G un graf neorientat cu n vârfuri dat prin matricea de adiacență. Să se realizeze un program care determină toate subgrafurile lui G . Matricele de adiacență ale subgrafurilor vor fi afișate într-un fișier text.
- Fiind date matricele de adiacență A și B de dimensiune n , să se verifice dacă matricea B poate reprezenta un graf parțial al grafului reprezentat de matricea A .
- Să se determine toate grafurile parțiale ale unui graf dat prin matricea de adiacență cu n vârfuri. Matricele de adiacență ale grafurilor parțiale vor fi afișate într-un fișier text.
- Fie G un graf neorientat cu n vârfuri și cu muchiile $m_1, m_2, \dots, m_p$. Să se realizeze un program care determină toate grafurile parțiale ale lui G .
- Se citesc 2 grafuri neorientate, unul cu n vârfuri și m muchii, iar celălalt cu k vârfuri și l muchii, ambele date prin vectorul muchiilor. Să se determine dacă al doilea graf este subgraf al primului.
- Se consideră lista muchiilor unui graf neorientat cu n vârfuri, etichetate de la 1 la n . Din acest graf se elimină toate vârfurile etichetate cu valori pare. Să se determine câte muchii va avea subgraful obținut.

9. Se consideră lista muchiilor unui graf neorientat cu n vârfuri, etichetate de la 1 la n . Din acest graf se elimină toate vârfurile etichetate cu valori prime. Să se determine câte muchii va avea subgraful obținut.
10. Se citește de la tastatură un graf neorientat cu n vârfuri și m muchii, dat prin matricea de adiacență. Fie k un număr natural mai mic decât n . Să se precizeze câte subgrafuri cu k vârfuri ale grafului inițial sunt complete.
11. Se consideră lista muchiilor unui graf neorientat cu n vârfuri, etichetate de la 1 la n . Din acest graf se elimină toate muchiile care au ca extremități 2 numere impare. Să se determine câte muchii va avea graful parțial obținut.
12. Se consideră lista muchiilor unui graf neorientat cu n vârfuri, etichetate de la 1 la n . Din acest graf se elimină toate muchiile care au ca extremități 2 numere prime. Să se determine câte muchii va avea graful parțial obținut.
13. Se consideră lista muchiilor unui graf neorientat cu n vârfuri, etichetate de la 1 la n . Din acest graf se elimină toate vârfurile care au gradul minim. Să se determine câte muchii va avea subgraful obținut.
14. Se consideră lista muchiilor unui graf neorientat cu n vârfuri, etichetate de la 1 la n . Din acest graf se elimină toate muchiile cu o extremitate într-un vârf de grad maxim. Să se determine numărul de muchii eliminate și să se afișeze matricea de adiacență a grafului parțial obținut.
15. Pentru un concurs de explorare, organizatorii au pregătit un circuit alcătuit din n puncte de control, numerotate 1,2,..., n prin care trebuie să treacă participanții în timpul concursului. Între unele dintre aceste puncte există drumuri directe, iar între altele nu. În urma evaluării traseului de către o comisie, s-a constatat că circuitul propus de organizatori este prea ușor, motiv pentru care comisia a solicitat eliminarea din circuit a acelor puncte de control în care se poate ajunge pe un drum direct din toate celelalte puncte. Circuitul inițial este memorat sub forma unei matrice a în care elementul a_{ij} este 1 dacă există drum direct între punctele i și j și 0 în caz contrar. Să se afișeze circuitul obținut în urma eliminării cerute de comisie. Matricea inițială se va citi dintr-un fișier text.

7.1.3. Lanțuri și cicluri

1. Fiind dată o succesiune de vârfuri $x_1, x_2, \dots, x_p$, să se verifice dacă poate reprezenta un lanț pentru un graf neorientat $G = (X, U)$, dat prin matricea de adiacență.
2. Se dă lista muchiilor unui graf neorientat cu n vârfuri și două vârfuri p, q . Să se determine toate lanțurile elementare cu extremitățile p și q .
3. Fiind dată lista muchiilor unui graf neorientat cu n vârfuri și două vârfuri p, q . Să se determine cel mai lung lanț elementar cu extremitățile p și q .
4. Să se verifice dacă o succesiune de vârfuri date este ciclu pentru un graf. În caz afirmativ, să se stabilească dacă este ciclu elementar sau nu.
5. Într-un graf neorientat G se dau două vârfuri a și b . Să se verifice dacă în graf există un ciclu care să conțină vârfurile a și b .
6. Fiind date un graf neorientat G și două vârfuri x și y ale sale, să se determine cel mai lung (cel mai scurt) lanț elementar având ca extremități vârfurile date.
7. Să se realizeze un program de generare a tuturor lanțurilor unui graf. Rezultatul va fi trecut într-un fișier text.
8. Să se realizeze un program de generare a tuturor ciclurilor unui graf. Rezultatul va fi trecut într-un fișier text.
9. Fiind dat un graf $G = (X, U)$, să se determine un lanț elementar de lungime maximă.
10. Fiind dat un graf neorientat cu n vârfuri și m muchii, să se determine toate lanțurile elementare care au lungime maximă.
11. Se consideră un graf neorientat cu n vârfuri și m muchii dat prin vectorul de muchii. Să se calculeze distanțele minime de la un vârf citit la toate celelalte vârfuri ale grafului. Distanța minimă dintre două vârfuri este egală cu lungimea celui mai scurt lanț care are cele două vârfuri ca extremități.
12. Fiind dat un graf neorientat, să se determine lungimea lanțului minim între două vârfuri x și y citite de la tastatură.
13. Se consideră lista muchiilor unui graf neorientat cu n vârfuri și trei vârfuri p, q și r . Să se determine toate lanțurile elementare cu extremitățile în p și q care conțin vârful r .
14. Se dă lista muchiilor unui graf neorientat cu n vârfuri și un vârf p . Să se determine un ciclu elementar care conține vârful p .
15. Să se scrie un program care să decidă dacă într-un graf neorientat dat există un ciclu de lungime patru.
16. Se citește de la tastatură un graf neorientat cu n vârfuri și m muchii, dat prin vectorul de muchii. Să se elimine un număr minim de muchii din graf, astfel încât acesta să nu mai conțină nici un ciclu. Se vor afișa muchiile eliminate și matricea de adiacență a grafului parțial astfel obținut. Rezultatele vor fi trecute într-un fișier text.
17. Să se testeze dacă un graf neorientat, dat prin lista muchiilor sale, este aciclic.
18. Fie G un graf neorientat dat prin matricea de adiacență. Să se verifice dacă graful neorientat astfel introdus este aciclic.
19. Fiind dat un graf neorientat $G = (X, U)$, să se determine toate perechile de vârfuri între care există cel puțin un lanț. Matricea de adiacență a grafului se va citi dintr-un fișier text.

20. Fișierul "Graf.in" conține pe primul rând un număr natural n , pe fiecare din următoarele n rânduri elementele unei linii a matricei de adiacență asociată grafului, separate prin spații. Să se genereze prin metoda backtracking toate ciclurile elementare din graf. Rezultatul va fi afișat în fișierul "Graf.out".
21. Din fișierul "Graf.in" se citesc de pe prima linie numerele naturale n și p , reprezentând numărul de vârfuri ale unui graf complet și respectiv lungimea unui ciclu. Să se afișeze în fișierul "Graf.out" toate ciclurile de lungime p ale grafului complet cu n vârfuri.
22. Algoritmul Roy-Warshall. Se citește un graf neorientat cu n vârfuri și m muchii dat prin vectorul de muchii. Să se construiască matricea lanțurilor ($a_{ij} = 1$ dacă există lanț de la i la j și 0 în caz contrar.)
23. Fie $G(X, U)$ un graf neorientat cu n vârfuri și m muchii, unde X este mulțimea vârfurilor, iar U este mulțimea muchiilor. Numim *diametrul grafului*, notat $D(G)$ valoarea expresiei $\max\{l[i] \mid i \in X\}$, unde $l[i]$ este distanța maximă, care are ca extremitate inițială vârful i . Pentru un graf dat prin matricea sa de adiacență, să se calculeze diametrul său.

7.1.4. Parcurgerea grafurilor neorientate

1. Se consideră un graf neorientat cu n vârfuri și m muchii și un vârf cunoscut x . Să se afișeze vârfurile vizitate în urma parcurgerii în lățime a grafului, pornind din vârful x .
2. Se consideră un graf neorientat cu n vârfuri și m muchii și un vârf x . Să se afișeze vârfurile vizitate în urma parcurgerii în adâncime a grafului, pornind din vârful x .
3. Fie G un graf neorientat cu n vârfuri și m muchii. Să se realizeze un program care afișează toate vârfurile accesibile dintr-un vârf x dat.
4. Se consideră lista muchiilor unui graf neorientat. Să se verifice dacă graful este sau nu conex.
5. Se dă lista muchiilor unui graf neorientat. Să se afișeze componentele conexe ale acestui graf.
6. Se dă lista muchiilor unui graf neorientat conex cu n vârfuri, etichetate de la 1 la n . Să se verifice dacă graful este bipartit.
7. Se dă un graf neorientat cu n vârfuri. Determinați, dacă există, un ciclu hamiltonian. În cazul în care nu există se va afișa un mesaj corespunzător.
8. Se dă un graf neorientat cu n vârfuri care este conex și are gradele tuturor vârfurilor pare. Determinați un ciclu eulerian. În cazul în care nu există se va afișa un mesaj corespunzător.
9. Se dă lista muchiilor unui graf neorientat și două vârfuri p și q . Să se determine cel mai scurt lanț cu extremitățile p, q .
10. Se dă lista muchiilor unui graf neorientat cu n vârfuri și un vârf p . Să se determine toate nodurile q ale grafului cu proprietatea că lungimea minimă a unui lanț de la q la p este L .
11. Se dă lista muchiilor unui graf neorientat și trei vârfuri p, q și r . Să se determine un lanț cu extremitățile p, q care conține vârful r .
12. Fie G un graf neorientat cu n vârfuri și m muchii. Să se determine cea mai mare distanță dintre vârful 1 și celelalte vârfuri.
13. Fie G un graf neorientat conex. Să se determine componentele biconexe, punctele de articulație și muchiile critice ale acestuia.
14. Se consideră un graf neorientat conex cu n vârfuri, numerotate de la 1 la n , și m muchii. Definim distanța minimă dintre două noduri x și y ca fiind numărul minim de muchii al unui lanț elementar care unește x cu y . Se dau k perechi de vârfuri x, y . Determinați pentru fiecare pereche distanța minimă de la x la y .
15. Harta rutieră a unei țări conține toate autostrăzile care leagă perechi de orașe x și y . Să se afișeze toate orașele la care se poate ajunge dintr-un oraș a , folosind p autostrăzi.
16. Se consideră un graf neorientat cu n vârfuri și m muchii dat prin vectorul de muchii. Să se construiască matricea lanțurilor, folosind parcurgerea în adâncime.
17. Se consideră un graf neorientat G , dat prin matricea de adiacență. Să se verifice dacă succesiunea vârfurilor la parcurgerea în adâncime și lățime este aceeași, pornind dintr-un vârf x dat.
18. Să se scrie un program care pentru un graf dat $G = (X, U)$ cu n vârfuri și un număr natural k , determină subgraful $H = (Y, V)$ cu un număr maxim de vârfuri și cu proprietatea că orice vârf al său are gradul cel puțin k (se va afișa un mesaj corespunzător în cazul în care un asemenea subgraf nu există).
19. Fișierul "Graf.txt" conține pe primul rând un număr întreg n , apoi pe fiecare din următoarele n rânduri, elementele unei linii a matricei de adiacență pentru un graf neorientat cu n vârfuri și m muchii (separate prin spații). Scrieți un program, care, folosind parcurgerea DF (în adâncime) verifică dacă graful este conex sau nu. Dacă graful nu este conex va afișa numărul de componente conexe și mulțimile de vârfuri care le generează.
20. Se dă un graf neorientat G conex. Să se determine arborii obținuți în urma parcurgerii BF , respectiv DF , pornind dintr-un vârf dat.
21. Se consideră un graf neorientat G cu n vârfuri și m muchii. Să se determine numărul minim de culori cu care se pot colora vârfurile grafului, astfel încât oricare două vârfuri adiacente să aibă culori diferite. Culorile sunt codificate prin numere. Folosind această codificare, să se realizeze o colorare a vârfurilor grafului dat. Să se afișeze pe câte un rând vârfurile care se colorează cu o anumită culoare.

22. Un sistem de transport feroviar constă din m linii de transport. În toată țara există n stații CFR, pe care le considerăm numerotate de la 1 la n . Fiecare linie de transport constă dintr-o succesiune de stații și poate fi parcursă în ambele sensuri. Să se scrie un program care să determine numărul minim de schimbări de pe o linie pe alta, necesar pentru a ajunge de la stația i la stația j . Schimbarea de linie este posibilă doar într-o stație prin care trec ambele linii.
- Datele de intrare se citesc din fișierul "Transport.in" astfel: pe prima linie se găsesc patru valori n, m, i, j cu semnificațiile prezentate mai sus, pe următoarele m rânduri stațiile aferente fiecărei linii. Fișierul de ieșire "Transport.out" va conține o singură valoare, reprezentând numărul de schimbări necesar pentru a ajunge din stația i în stația j , trecând prin schimbări de linii.
23. Pentru un concurs de explorare s-au stabilit n puncte de control, numerotate de la 1 la n . În concurs există m perechi de puncte de control între care există legături. Între oricare două puncte de control există legături fie directe, fie prin intermediul altor puncte. Să se stabilească, pentru fiecare dintre cele n puncte de control, dacă eliminarea sa de pe harta concursului, conduce la pierderea legăturii dintre cel puțin alte două puncte de control rămase.
24. Pe un grup de socializare s-au înscris n persoane. Anumite perechi de persoane se cunosc între ele. Aceste perechi de persoane prietene sunt cunoscute. Relația de prietenie între două persoane este reciprocă. Într-o zi, p dintre cele n persoane se hotărăsc să distribuie un mesaj, care să fie distribuit mai departe și celorlalte persoane, în decurs de o zi. Să se determine după câte zile mesajul ajunge la toate cele n persoane.
25. Ionuț tocmai a terminat liceul și susține examenul de admitere la facultate. Știind că s-a pregătit foarte bine pentru examen, el dorește să își anunțe reușita după examen printr-o postare pe Facebook.
- Ionuț cunoaște n utilizatori reprezentați de numerele de la 1 la n , între care există m relații de prietenie de forma i, j , unde i și j sunt utilizatori, iar n și m sunt numere naturale nenule. Un utilizator nu poate fi prieten cu el însuși, iar o relație de prietenie între doi utilizatori ne spune că fiecare dintre ei este prieten cu celălalt.
- Întrucât dorește ca postarea lui să fie cât mai răspândită, Ionuț vrea să afle care sunt utilizatorii cei mai bine conectați din mulțimea sa de cunoscuți, pentru ca eventual să le ceară prietenia. Pentru aceasta, Ionuț trebuie să găsească cea mai mare submulțime de utilizatori cunoscuți, în care fiecare utilizator din această submulțime are cel puțin k prieteni aflați la rândul lor în submulțime, unde k este un număr natural nenul.
- Ajutați-l pe Ionuț să se determine și să se afișeze, printr-o soluție de complexitate timp cât mai bună, în funcție de datele de intrare, membrii celei mai mari submulțimi de utilizatori, cu proprietatea că fiecare utilizator din această submulțime are cel puțin k prieteni aflați la rândul lor în submulțime. (Admitere FMI București - 2016)

7.1.5. Conexitate. Componente conexe

1. Scrieți un program care determină dacă un graf neorientat dat prin matricea de adiacență este conex.
2. Scrieți un program care determină dacă un graf neorientat dat prin vectorul de muchii este conex.
3. Scrieți un program care determină componentele conexe ale unui graf neorientat dat.
4. Se dă un graf neorientat conex cu n vârfuri și număr par de muchii. Să se determine un graf parțial al celui dat care să fie conex și să fie obținut prin eliminarea a jumătate din numărul de muchii.
5. Fiind dat un graf neorientat conex $G = (X, U)$, să se determine un vârf $x \in X$, astfel încât subgraful indus de mulțimea de vârfuri $X \setminus \{x\}$ să fie conex.
6. Se dau graful conex $G = (X, U)$ și un vârf $x \in X$. Să se verifice dacă subgraful indus de mulțimea de vârfuri $X \setminus \{x\}$ este conex.
7. Pentru un graf neconex, să se adauge un număr minim de muchii astfel încât graful să devină conex. Să se afișeze muchiile adăugate într-un fișier text.
8. Pentru un graf conex cu n vârfuri și un număr natural k dat ($1 \leq k \leq n$), să se construiască un subgraf conex $H = (Y, V)$ având k vârfuri. Se vor preciza vârfurile și muchiile subgrafului determinat.
9. Să se determine componentele conexe ale unui graf neorientat folosind matricea lanțurilor.
10. Se dau un graf conex $G = (X, V)$ și $S \subset X$, $\text{card}(S) \geq 2$. Să se elimine un număr minim de muchii astfel încât fiecare vârf din S să fie într-o componentă conexă disjunctă. Să se precizeze muchiile eliminate.
11. Se dă lista muchiilor unui graf neorientat. Să se determine numărul de muchii care pot fi eliminate din graf astfel încât numărul de componente conexe ale grafului să nu se modifice.
12. Se citește de la tastatură un graf neorientat cu n vârfuri și m muchii, citit prin vectorul de muchii. Să se afișeze cea mai mare dintre componentele sale conexe.
13. Se consideră un graf neorientat cu n vârfuri și m muchii. Să se precizeze la fiecare eliminare a câte unei muchii din graf, câte componente conexe va avea graful. La sfârșit vor fi eliminate toate cele m muchii.
14. Se consideră un graf neorientat cu n vârfuri, definit prin listele vecinilor, în fișierul "Graf.txt". Fișierul conține pe primul rând valoarea lui n , apoi pe fiecare din următoarele n rânduri găsindu-se lista vecinilor unui vârf (elementele de pe un rând sunt separate prin spații). Să se afișeze toate componentele conexe ale grafului, precum și numărul lor.

15. Se citește de la tastatură un graf neorientat cu n vârfuri și m muchii dat prin vectorul de muchii. Să se construiască matricea lanțurilor folosind algoritmul *Roy-Warshall*. Folosind matricea astfel construită să se afișeze componentele conexe.
16. O echipă de speologi a cercetat n culoare subterane, pentru a stabili dacă aparțin aceleiași peșteri. Prin tehnici specifice bine stabilite anterior, a fost demonstrată existența unor canale de legătură între mai multe culoare. Precizându-se perechile de culoare între care au fost stabilite legături, să se afle dacă sistemul de culoare aparține unei singure peșteri.
17. Într-un grup de n persoane se precizează perechi de persoane care se consideră prietene. Folosind principiul că "*prietenul prietenului meu îmi este prieten*", să se determine grupurile cu un număr maxim de persoane între care se pot stabili relații de prietenie, directe sau indirecte.
18. Să se determine câte grupuri de prieteni există într-o școală și în câte moduri se poate forma o nouă relație de prietenie, astfel încât, să se obțină un nou grup de prieteni, cu număr maxim de membri.
19. Între elevii dintr-o școală, identificați prin nume, există relații de prietenie, precizate prin perechi de forma (Horia, Andreea), cu semnificația "Horia este prieten cu Andreea". Să se stabilească numărul elevilor, știind că fiecare are cel puțin un prieten. Să se afișeze numărul de grupe de prieteni și componența acestora. Relația de prietenie este simetrică și tranzitivă. Rezultatele vor fi afișate într-un fișier text.
20. Fișierul text "*Cuvinte.in*" conține m perechi de cuvinte înrudite (de forma (măr, păr) sau (pom, păr)). Fiecare pereche se găsește pe câte un rând din fișier. Cuvintele unei perechi sunt separate prin spații. Să se găsească grupurile de cuvinte înrudite și să se afișeze în fișierul "*Cuvinte.out*". Cuvintele dintr-un grup se vor afișa pe un rând în fișier și vor fi separate prin spații.

7.1.6. Tipuri de grafuri speciale

1. Se consideră un graf neorientat reprezentat prin matricea de adiacență. Să se verifice dacă graful este complet.
2. Se consideră un graf neorientat reprezentat prin vectorul de muchii. Să se verifice dacă graful este complet.
3. Se consideră un graf neorientat reprezentat prin matricea de adiacență. Să se afișeze ce muchii trebuie adăugate ca graful să devină complet.
4. Se consideră un graf neorientat reprezentat prin vectorul de muchii. Să se afișeze ce muchii trebuie adăugate ca graful să devină complet.

5. Se consideră lista muchiilor unui graf neorientat cu n vârfuri, etichetate de la 1 la n , precum și o mulțime A de vârfuri ale grafului. Considerăm mulțimea B formată din vârfurile grafului care nu aparțin lui A . Să se verifice dacă graful este bipartit peste partiția formată din mulțimile A și B .
6. Se consideră lista muchiilor unui graf neorientat cu n vârfuri, etichetate de la 1 la n . Să se verifice dacă graful este bipartit. În caz afirmativ să se afișeze mulțimile A și B .
7. Să se verifice dacă un graf neorientat este regulat de grad k , unde k este citit de la tastatură. Graful este dat prin matricea de adiacență (un graf este regulat dacă toate vârfurile sale au același grad).
8. Se consideră două mulțimi nevide A și B , cu proprietatea că formează o partiție a mulțimii $\{1, 2, \dots, n\}$. Să se construiască un graf bipartit complet cu n vârfuri, peste partiția formată din mulțimile A și B .
9. Fiind dat un număr natural n , ($n \geq 2$) să se determine toate grafurile bipartite complete cu n vârfuri.
10. Să se genereze un graf neorientat complet de ordin n . Matricea de adiacență asociată grafului astfel generat va fi afișată într-un fișier.
11. Fiind dat un graf neorientat, să se verifice dacă este graf hamiltonian și în caz afirmativ să se determine un ciclu hamiltonian al său.
12. Să se determine toate lanțurile hamiltoniene dintr-un graf neorientat dat.
13. Fiind dat un graf neorientat, să se verifice dacă este graf eulerian și în caz afirmativ să se determine un ciclu eulerian al său.
14. Se dă un graf neorientat cu n vârfuri. Determinați, dacă există, un ciclu hamiltonian. În caz afirmativ, să se afișeze într-un fișier text.
15. Se consideră un graf neorientat cu n vârfuri care este conex și are gradele tuturor vârfurilor pare. Determinați un ciclu eulerian.
16. Se consideră un graf neorientat cu n vârfuri și m muchii, citit de la tastatură prin vectorul de muchii. Să se afișeze toate ciclurile euleriene pe care le are graful. Ciclurile euleriene astfel determinate vor fi afișate în fișierul "*Eulerian.txt*".
17. Aflați numărul de cicluri hamiltoniene dintr-un graf neorientat complet cu n noduri.
18. Fie G un graf complet cu n vârfuri în care muchiile sunt colorate în *alb* și *negru*. În acest graf se pot forma triunghiuri monocromatice (trei vârfuri unite prin muchii de aceeași culoare). Să se scrie un program care să determine pentru un graf colorat dat, numărul de triunghiuri monocromatice.
19. Fiind dată matricea de adiacență a unui graf neorientat conex cu n vârfuri, se cere să se adauge un număr minim de muchii astfel încât graful să devină eulerian. Dacă acest lucru nu este posibil, se va afișa un mesaj corespunzător.

20. La curtea regelui Arthur s-au adunat $2n$ cavaleri și fiecare dintre ei are printre cei prezenți cel mult $n - 1$ dușmani. Merlin, consilierul lui Arthur, îi așează pe cavaleri la o masă rotundă, astfel încât nici unul dintre ei să nu stea alături de vreun dușman de-al său. Determinați așezarea propusă de Merlin.
21. Se consideră n persoane, date prin numerele lor de ordine $1, 2, 3, \dots, n$. Fiecare are printre cei prezenți $n/2$ dușmani. Să se determine toate posibilitățile de așezare a acestora la o masă rotundă astfel încât nici unul dintre ei să nu stea lângă un dușman de-al său.
22. La un concurs școlar, sunt invitați un număr de n elevi, identificați prin numerele lor de ordine $1, 2, \dots, n$ de la C colegii identificate cu $1, 2, \dots, C$. Să se determine toate modalitățile de așezare a elevilor la o masă rotundă, astfel încât să nu fie doi elevi de la același colegiu, vecini.
23. Elevii din clasa a XI-a și-au format un sistem propriu de transmitere a fotografiilor pe telefon. În urma unei ședințe la care au participat cu toții, au stabilit un sistem de relații, astfel încât oricare ar fi doi elevi, unul poate primi fotografii de la celălalt. De asemenea, orice fotografie trimisă poate fi transmisă mai departe. Presupunând că în clasa a XI-a sunt n elevi, numerotați de la 1 la n și că sistemul de relații dintre ei este cunoscut, să se scrie un program care să găsească o modalitate prin care o fotografie, transmisă de unul dintre elevi să ajungă pe la toți elevii o singură dată.
24. Un curier de la o firmă de curierat și-a propus să-și optimizeze traseul pe care trebuie să-l parcurgă zilnic pentru distribuirea mărfii. În fiecare zi el pleacă de la sediu și trebuie să se întoarcă la sediu. În cartierul unde trebuie să distribuie produsele sunt m străzi. Soluția pentru optimizarea traseului stabilită de el este următoarea: numerează intersecțiile, intersecția numărul 1 fiind intersecția în care se găsește sediul. Se poate considera intersecție și un capăt de stradă. Pentru a-și putea finaliza proiectul, a conceput o hartă pe care a marcat intersecțiile și străzile. În cartier pot exista străzi distincte, care au ambele capete în aceleași intersecții. De la sediu poate ajunge pe orice stradă dorește, trecând doar pe străzi din cartier. Curierul dorește un traseu astfel încât să nu fie nevoit să treacă de două ori pe aceeași stradă. Scrieți un program care să determine un traseu convenabil pentru curier. Fișierul "Curier.in" va conține pe prima linie două valori n și m , iar pe următoarele m linii, extremitățile tuturor străzilor. Fișierul "Curier.out" va conține pe prima linie intersecțiile în ordinea în care sunt parcurse pe un traseu optim (să nu treacă de două ori pe aceeași stradă). Traseul va începe cu 1 și se va termina cu 1, pentru că pleacă de la sediu și se întoarce la sediu. În cazul în care problema nu are soluție, se va afișa un mesaj corespunzător în fișierul de ieșire.

7.2. Grafuri orientate (Digrafuri)

7.2.1. Noțiuni generale

1. Să se memoreze și să se afișeze un graf orientat folosind cele patru modalități de reprezentare:
 - Matrice de adiacență;
 - Vector de arce;
 - Liste de vecini;
 - Matrice de incidență.
2. Fie G un graf orientat, cu n noduri și m arce, reprezentat prin matricea de adiacență. Să se scrie un program care afișează vectorul de muchii asociat grafului dat.
3. Fie G un graf orientat, cu n noduri și m arce, reprezentat prin matricea de adiacență. Să se scrie un program care afișează listele de vecini asociate grafului dat.
4. Fie G un graf orientat, cu n noduri și m arce, reprezentat prin matricea de incidență. Să se scrie un program care afișează matricea de adiacență asociată grafului dat.
5. Fie G un graf orientat, cu n vârfuri, reprezentat prin matricea de adiacență. Să se scrie un program care afișează gradul exterior și gradul interior al unui nod dat.
6. Fie G un graf orientat, cu n vârfuri, reprezentat prin matricea de adiacență. Să se afișeze gradele tuturor nodurilor (gradele interioare și exterioare). Rezultatul va fi trecut în fișierul text "Grade.txt".
7. Se dă lista muchiilor unui graf orientat. Să se afișeze gradele tuturor nodurilor (gradele interioare și exterioare). Rezultatul va fi trecut în fișierul text "Grade.txt".
8. Fie G un graf orientat, cu n vârfuri, reprezentat prin matricea de adiacență. Să se afișeze nodurile care au gradul exterior număr par.
9. Fie G un graf orientat, cu n noduri, reprezentat prin matricea de adiacență. Să se afișeze nodurile izolate (nodurile cu gradul 0). Dacă nu are noduri izolate, se va afișa un mesaj corespunzător.
10. Fie G un graf orientat, cu n noduri, reprezentat prin matricea de adiacență. Să se afișeze nodurile terminale (nodurile cu gradul 1).
11. Fie G un graf orientat, cu n noduri, reprezentat prin matricea de adiacență. Să se scrie un program care verifică dacă graful are toate nodurile interioare (nu sunt nici izolate, nici terminale).
12. Fie G un graf orientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se scrie un program care afișează listele vecinilor tuturor nodurilor.

13. Se dă lista arcelor unui graf orientat. Să se determine nodurile care au gradul exterior egal cu gradul interior.
14. Se citește matricea de adiacență a unui graf orientat. Să se afișeze toate nodurile pentru care $d^+(x) = d^-(x)$ (gradul exterior este egal cu gradul interior).
15. Se dă lista arcelor unui graf orientat. Să se determine nodurile care au gradul interior nul.
16. Se citește matricea de adiacență a unui graf orientat. Pentru un nod x citit de la tastatură, să se afișeze toate nodurile adiacente cu acesta.
17. Fie G un graf orientat, cu n noduri. Numim *matricea costurilor* (ponderilor) pentru un graf orientat, matricea pentru care $a_{ij} = c$ unde c reprezintă costul muchiei (i, j) și 0 în cazul în care nu există muchie (i, j) . Să se reprezinte un graf prin matricea costurilor.
18. Se citește de la tastatură un vector de înregistrări, reprezentând vectorul de arce al unui graf orientat G . Fiecare înregistrare va avea trei câmpuri: extremitățile unui arc și costul asociat lui. Să se construiască matricea costurilor asociată grafului.
19. Se definește un arc al unui graf orientat ca o înregistrare cu trei câmpuri: extremitățile unui arc și un cost asociat lui (un număr întreg). Definim un graf orientat ca un vector de arce, fiecare element al vectorului având forma descrisă mai sus. Să se afișeze arcul (arcele) de *cost minim*.
20. Fie G un graf orientat cu n noduri, reprezentat prin matricea costurilor. Să se afișeze arcul (arcele) de *cost minim*.
21. Se definește un arc al unui graf orientat ca o înregistrare cu trei câmpuri: extremitățile arcului și un cost asociat lui (un număr întreg). Definim un graf orientat ca un vector de arce, fiecare element al vectorului având forma descrisă mai sus. Să se afișeze *costul mediu* al grafului (media aritmetică a costurilor muchiilor).
22. Fie G un graf orientat cu n noduri, reprezentat prin matricea costurilor. Să se afișeze *costul mediu* al grafului (media aritmetică a costurilor muchiilor).
23. Fie G un graf orientat, cu n noduri, reprezentat prin matricea de adiacență. Fiind dat șirul $d_1, d_2, \dots, d_n$, să se verifice dacă poate reprezenta șirul gradelor interioare ale nodurilor, în această ordine.
24. Fie G un graf orientat, cu n noduri, reprezentat prin matricea de adiacență. Fiind dat șirul $d_1, d_2, \dots, d_n$, să se verifice dacă poate reprezenta șirul gradelor exterioare ale nodurilor, nu neapărat în această ordine.
25. Se citește secvența de numere naturale $d_1 \geq d_2 \geq d_3 \geq \dots \geq d_n$. Să se decidă dacă această *secvență* este *grafică*, adică dacă există un graf orientat cu n noduri ale căror grade exterioare să fie chiar numerele $d_1 \geq d_2 \geq d_3 \geq \dots \geq d_n$.

26. Se citește un graf orientat de la tastatură dat prin matricea de adiacență. Să se afișeze noduri incidente cu arcul (x, y) din graf.
27. Se citește un graf orientat de la tastatură dat prin matricea de adiacență. Să se afișeze muchiile incidente cu nodul x din graf.
28. Se citește un graf orientat de la tastatură dat prin listele vecinilor. Să se afișeze nodul (nodurile) cu cei mai mulți vecini.
29. Se dă matricea de adiacență a unui graf orientat. Să se afișeze, pentru fiecare nod al grafului, lista vecinilor săi. Rezultatul va fi trecut în fișierul "*Vecini.txt*", fiecare nod având vecinii pe câte un rând în fișier.
30. Se dă lista arcelor unui graf orientat. Să se afișeze nodurile de grad exterior maxim.
31. Se consideră matricea de adiacență a unui graf orientat. Să se afișeze nodurile de grad interior maxim.
32. Fișierul "*Graf.in*" conține pe prima linie numărul n de noduri și numărul de arce ale unui graf orientat G , iar pe următoarele n linii, elementele din matricea de adiacență atașată grafului. Să se afișeze în fișierul "*Graf.out*" nodurile al căror grad interior se află într-un interval dat $[x, y]$ (x și y sunt numere naturale introduse de la tastatură).
33. Fișierul "*Graf.in*" conține pe primul rând două valori n și m , reprezentând numărul de noduri, respectiv numărul de arce ale unui graf orientat, apoi pe fiecare dintre cele m rânduri câte o pereche de numere întregi separate prin spații, reprezentând extremitățile unui arc. Să se scrie în fișierul "*Graf.out*" nodurile grafului, în ordinea descrescătoare a gradelor, apoi să se afișeze câte dintre nodurile grafului au același grad.
34. Se dă lista arcelor unui graf orientat cu n noduri, etichetate de la 1 la n . Să se elimine toate arcele cu proprietatea că ambele extremități au aceeași paritate. Să se afișeze graful astfel obținut.
35. Se dă matricea de adiacență a unui graf orientat cu n noduri, etichetate de la 1 la n . Să se elimine toate arcele cu proprietatea că extremitățile lor au parități diferite. Să se afișeze graful astfel obținut.
36. Fiind dată o matrice pătratică de ordin n , să se precizeze dacă poate fi considerată matricea de adiacență a unui graf orientat cu n vârfuri.
37. Fiind dat un vector în care fiecare înregistrare are două câmpuri, reprezentând extremitățile câte unui arc, să se precizeze dacă poate fi considerat vectorul de arce al unui graf orientat cu n noduri.
38. Se citește din fișierul "*Graf.in*" de pe prima linie numărul de noduri și numărul de arce n , respectiv m , de pe următoarele m rânduri, perechi de numere reprezentând arcele grafului. Să se construiască matricea de adiacență și să se scrie aceasta în fișierul "*Graf.out*".
39. Se citește din fișierul "*Graf.in*" de pe prima linie numărul de noduri și numărul de arce (n, m) , de pe următoarele m rânduri, perechi de noduri reprezentând arcele grafului. Să se calculeze gradele interioare și exterioare pentru fiecare nod și să se memoreze în doi vectori. Vectorii astfel obținuți se vor afișa în fișierul "*Graf.out*".

40. Se citește din fișierul "Graf.in" de pe prima linie numărul de noduri și numărul de arce (n, m) , de pe următoarele m rânduri, perechi de noduri reprezentând arcele grafului. Să se calculeze frecvențele gradelor interioare și exterioare și să se memoreze într-un vector de înregistrări. Vectorul astfel obținut se va afișa în fișierul "Graf.out", fiecare înregistrare pe câte un rând.
41. Să se coloreze în toate modurile posibile muchiile unui graf orientat folosind c culori codificate $1, 2, \dots, c$, dacă acest lucru este posibil. Orice două muchii incidente trebuie să fie colorate diferit.
42. Se dă un graf orientat cu n noduri și m arce prin lista arcelor. Se citesc apoi un număr k și o submulțime X cu k vârfuri din mulțimea vârfurilor grafului, notată cu V . Afișați arcele din graful citit, care au proprietatea că au o extremitate în mulțimea X și cealaltă în mulțimea $V \setminus X$.
43. Într-un grup sunt n persoane, numerotate de la 1 la n . O persoană poate cunoaște alte persoane – relație care nu este reciprocă. Să se determine persoana cea mai cunoscută.
44. Se dă un grup format din n persoane, care se cunosc sau nu între ele. De la tastatură se introduc m perechi de numere întregi (x, y) cu semnificația "persoana x cunoaște persoana y ". Relația de cunoștință nu este neapărat reciprocă. Numim *celebritate*, o persoană care este cunoscută de către toate celelalte persoane din grup, dar ea nu cunoaște pe nici un alt membru al grupului. Să se determine dacă în grup există o astfel de celebritate.
45. Se consideră un graf orientat cu n noduri și m arce, dat prin lista arcelor. Determinați numărul minim de culori cu care se pot colora nodurile grafului astfel încât oricare două noduri adiacente să aibă culori diferite. Fiecare culoare este codificată printr-un număr de la 1 la m (m este numărul minim de culori). Să se realizeze o colorare a nodurilor grafului, precizându-se pentru fiecare nod culoarea cu care s-a colorat. De asemenea se vor afișa pentru fiecare culoare nodurile corespunzătoare. Rezultatele vor fi afișate în fișierul "Culori.txt".
46. Se dă un graf orientat aciclic cu n noduri numerotate de la 1 la n . Să se realizeze o *sortare topologică* a nodurilor. (O *sortare topologică* a nodurilor unui graf orientat aciclic este o operație de ordonare liniară a nodurilor, astfel încât, dacă există un arc (i, j) , atunci i apare înaintea lui j în această ordonare.)

7.2.2. Grafuri orientate asociate unui graf dat

1. Să se verifice dacă un graf orientat dat prin matricea de adiacență A este subgraf al unui graf orientat dat prin matricea de adiacență B .
2. Să se verifice dacă un graf orientat dat prin matricea de adiacență A este graf parțial al unui graf orientat dat prin matricea de adiacență B .
3. Se citește de la tastatură un graf orientat cu n noduri dat prin matricea de adiacență. Fie k un număr natural mai mic decât n . Să se afișeze toate subgrafurile cu k noduri ale grafului dat. Subgrafurile vor fi date prin matricele de adiacență și vor fi afișate într-un fișier text.
4. Se citește de la tastatură un graf orientat cu n noduri și m arce, dat prin matricea de adiacență. Fie k un număr natural mai mic decât n . Să se afișeze cel mai mare subgraf cu k noduri al grafului dat (conține număr maxim de arce). Rezultatul va fi afișat într-un fișier text.
5. Se citește de la tastatură un graf orientat cu n noduri și m arce dat prin matricea de adiacență. Fie k un număr natural mai mic decât n . Să se precizeze câte subgrafuri cu k noduri ale grafului inițial sunt complete.
6. Se dau $G = (X, U)$ un graf orientat reprezentat prin matricea de adiacență și $x, y \in X, x \neq y$. Să se determine un subgraf al lui G cu număr minim de noduri, care conține pe x și y .
7. Fie G un graf orientat cu n noduri dat prin matricea de adiacență. Să se realizeze un program care determină toate subgrafurile lui G . Matricele de adiacență ale subgrafurilor vor fi afișate într-un fișier text.
8. Să se determine toate grafurile parțiale ale unui graf orientat, dat prin matricea de adiacență, cu n noduri. Matricele de adiacență ale grafurilor parțiale vor fi afișate într-un fișier text.
9. Fie G un graf orientat cu n noduri și p arce. Să se realizeze un program care determină toate grafurile parțiale ale lui G . Rezultatul va fi afișat într-un fișier text. Grafurile parțiale rezultate vor fi afișate prin vectorii de arce corespunzători.
10. Se consideră lista arcelor unui graf orientat cu n noduri, etichetate de la 1 la n . Din acest graf se elimină toate nodurile etichetate cu valori pare. Să se determine câte arce va avea subgraful obținut.
11. Se consideră lista arcelor unui graf orientat cu n noduri, etichetate de la 1 la n . Din acest graf se elimină toate nodurile etichetate cu valori prime. Să se determine câte arce va avea subgraful obținut.
12. Se citește de la tastatură un graf orientat cu n noduri și m arce, dat prin vectorul de arce. Fie k un număr natural mai mic decât n . Să se precizeze câte subgrafuri cu k noduri ale grafului inițial sunt complete.

13. Se consideră lista muchiilor unui graf orientat cu n noduri, etichetate de la 1 la n . Din acest graf se elimină toate arcele care au ca extremități 2 numere impare. Să se determine câte arce va avea graful parțial obținut.
14. Se consideră lista muchiilor unui graf orientat cu n noduri, etichetate de la 1 la n . Din acest graf se elimină toate arcele care au ca extremități 2 numere prime. Să se determine câte arce va avea graful parțial obținut.
15. Se consideră lista arcelor unui graf orientat cu n noduri, etichetate de la 1 la n . Din acest graf se elimină toate nodurile care au gradul minim. Să se determine câte arce va avea subgraful obținut.
16. Se consideră lista arcelor unui graf orientat cu n noduri, etichetate de la 1 la n . Din acest graf se elimină toate arcele cu o extremitate într-un nod de grad maxim. Să se determine numărul de arce eliminate și să se afișeze matricea de adiacență a grafului parțial obținut.

7.2.3. Drumuri și circuite

1. Fiind dată o succesiune de noduri $x_1, x_2, \dots, x_p$, să se verifice dacă poate reprezenta un drum pentru un graf orientat $G = (X, U)$, dat prin matricea de adiacență.
2. Fiind dată o succesiune de noduri $x_1, x_2, \dots, x_p$, să se verifice dacă poate reprezenta un circuit pentru un graf orientat $G = (X, U)$, dat prin matricea de adiacență. În caz afirmativ să se precizeze dacă este circuit elementar sau nu.
3. Se dă lista arcelor unui graf orientat cu n noduri și două vârfuri p, q . Să se determine toate drumurile elementare cu extremitățile p și q .
4. Fiind dată lista arcelor unui graf orientat cu n noduri și două noduri p, q , să se determine cel mai lung drum elementar cu extremitățile p și q .
5. Fiind dată matricea de adiacență a unui graf orientat cu n noduri și două noduri p, q , să se determine cel mai lung drum elementar cu extremitățile p și q .
6. Se dă mulțimea V a arcelor unui graf orientat $G(X, V)$ cu n vârfuri. Să se determine drumul de lungime maximă cu extremitatea inițială în vârful p , valoarea lui p este dată de la tastatură.
7. Se dă lista arcelor unui graf orientat. Construiți matricea drumurilor, folosind algoritmul *Roy-Warshall*.
8. Se consideră matricea de adiacență a unui graf orientat. Să se construiască matricea drumurilor, folosind algoritmul *Roy-Warshall*.
9. Se dă un graf orientat cu n noduri și un nod p . Să se afișeze toate nodurile q ale grafului, diferite de p , cu proprietatea că există cel puțin un drum de la p la q .
10. Se dă un graf orientat cu n noduri. Să se afișeze drumul (drumurile) de lungime maximă.
11. Se dă un graf orientat cu n noduri și un nod p . Să se afișeze toate nodurile q ale grafului, diferite de p , cu proprietatea că există cel puțin un drum de la p la q și lungimea drumului minim este o valoare pară.
12. Fie G un graf orientat cu n noduri. Să se verifice dacă în graf există un circuit care să conțină nodurile a și b (a și b sunt două numere naturale date).
13. Se dă lista arcelor unui graf orientat cu n noduri și un nod p . Să se determine un circuit elementar care conține nodul p .
14. Se citește de la tastatură un graf orientat cu n noduri și m arce dat prin matricea de adiacență. Se consideră un număr natural k mai mic decât n și k noduri ale grafului. Să se afișeze toate drumurile elementare care au ca extremitate inițială nodul 1, ca extremitate finală nodul n și care trec prin cele k noduri citite, în ordinea în care au fost citite.
15. Se citește un graf orientat cu n noduri și m arce, dat prin lista arcelor. Se citește apoi un număr natural k ($k < n$). Să se afișeze nodurile care se află la distanța k de nodul 1. (Distanța de la un nod x la un nod y este egală cu numărul de arce din care este compus cel mai scurt drum de la nodul x la nodul y).
16. Se citește de la tastatură un graf orientat cu n noduri și m arce, dar prin vectorul de arce. Să se determine dacă graful conține circuite. În caz afirmativ să se afișeze circuitele într-un fișier text.
17. Se citește un graf orientat cu n noduri și m arce dat prin lista arcelor. Să se calculeze lungimea minimă a drumului dintre oricare două noduri din graf.
18. Se citește de la tastatură un graf orientat cu n noduri și m arce, dat prin vectorul de arce. Să se elimine un număr minim de arce din graf, astfel încât acesta să nu mai conțină nici un circuit. Se vor afișa arcele eliminate și matricea de adiacență a grafului parțial astfel obținut. Rezultatele vor fi trecute într-un fișier text. Se consideră că graful are cel puțin un circuit.
19. Să se realizeze un program de generare a tuturor drumurilor unui graf orientat. Rezultatul va fi trecut într-un fișier text.
20. Să se realizeze un program de generare a tuturor circuitelor unui graf orientat. Rezultatul va fi trecut într-un fișier text.
21. Din fișierul "*Graf.in*" se citesc de pe prima linie numerele naturale n și p , reprezentând numărul de noduri ale unui graf complet și respectiv lungimea unui circuit. Să se afișeze în fișierul "*Graf.out*" toate circuitele de lungime p ale grafului complet cu n noduri.
22. Se dă lista arcelor unui graf orientat. Să se afișeze, în ordine lexicografică, toate circuitele de lungime k (k este dat). Circuitele astfel generate vor fi afișate în fișierul "*Circuite.txt*", câte unul pe fiecare rând.
23. Se consideră un graf orientat tare conex cu n noduri și m arce dat prin lista arcelor. Se numește *nod central* un nod cu proprietatea că suma distanțelor de la el la toate celelalte noduri este minimă. Distanța de la i la j este egală cu lungimea celui mai scurt drum de la i la j . Afișați nodurile centrale ale grafului.

7.2.4. Tare conexitate. Componente tare conexe

1. Să verifice dacă un graf orientat $G(X, U)$, dat prin matricea de adiacență este tare conex.
2. Scrieți un program care determină dacă un graf orientat dat prin vectorul de arce este tare conex.
3. Să se scrie un program care afișează componentele tare conexe ale unui graf orientat $G(X, U)$.
4. Se dă un graf orientat cu n noduri. Să se determine câte componente tare conexe are graful dat.
5. Se dă un graf orientat conex cu n vârfuri și număr par de arce. Să se determine un graf parțial al celui dat, care să fie tare conex și să fie obținut prin eliminarea a jumătate din numărul de arce. În cazul în care nu există, se va afișa un mesaj corespunzător.
6. Se consideră un graf orientat cu n vârfuri și m arce dat prin lista arcelor. Se numește *arc inutil*, un arc cu proprietatea că are extremitățile în componente tare conexe diferite. Afișați arcele inutile ale grafului dat.
7. Se consideră un graf orientat cu n noduri și m arce dat prin lista arcelor și un nod k . Să se afișeze toate nodurile din componenta tare conexă în care se află nodul k .
8. Se consideră un graf orientat cu n noduri și m arce dat prin lista arcelor și două noduri distincte a și b . Să se verifice dacă nodurile a și b se află în aceeași componentă tare conexă a grafului.
9. Fiind dat un graf orientat $G = (X, U)$, să se determine un nod $x \in X$, astfel încât subgraful indus de mulțimea de noduri $X \setminus \{x\}$ să fie tare conex. În cazul în care nu există, se va afișa un mesaj corespunzător.
10. Pentru un graf dat, să se adauge un număr minim de arce, astfel încât graful să devină tare conex. Să se afișeze arcele adăugate într-un fișier text.
11. Se consideră un graf orientat cu n noduri și m arce dat prin lista arcelor. Să se construiască matricea drumurilor grafului. Folosind matricea drumurilor, să se afișeze componentele tare conexe.
12. Se consideră un graf orientat cu n noduri și m arce dat prin matricea de adiacență. Să se afișeze nodurile care compun cea mai mare componentă tare conexă a grafului citit (cu cele mai multe vârfuri).
13. Se dau un graf tare conex $G = (X, V)$ și $Y \subset X$, $\text{card}(Y) \geq 2$. Să se elimine un număr minim de arce, astfel încât fiecare nod din Y să fie într-o componentă tare conexă disjunctă. Să se precizeze arcele eliminate.
14. Se dă lista arcelor unui graf orientat. Să se determine numărul de arce care pot fi eliminate din graf astfel încât numărul de componente tare conexe ale grafului să nu se modifice.
15. Se citește de la tastatură un graf orientat cu n noduri și m arce, citit prin vectorul de arce. Să se afișeze cea mai mare dintre componentele sale tare conexe.

16. Se consideră un graf orientat cu n noduri și m arce. Să se precizeze la fiecare eliminare a câte unui arc din graf, câte componente tare conexe va avea graful. La sfârșit vor fi eliminate toate cele m arce. Se vor afișa toate posibilitățile, într-un fișier text.
17. Se consideră un graf orientat cu n noduri, definit prin listele vecinilor, în fișierul "*Graf.in*". Fișierul conține pe primul rând valoarea lui n , apoi pe fiecare din următoarele n rânduri, găsindu-se lista vecinilor unui nod (elementele de pe un rând sunt separate prin spații). Să se afișeze în fișierul "*Graf.out*" toate componentele tare conexe ale grafului.

7.2.5. Tipuri speciale de grafuri

1. Să se verifice dacă un graf orientat dat prin matricea de adiacență este graf complet.
2. Dându-se lista arcelor unui graf orientat, să se verifice dacă graful este complet.
3. Se consideră un graf orientat, reprezentat prin matricea de adiacență. Să se afișeze ce arce trebuie adăugate ca graful să devină complet.
4. Se consideră un graf orientat cu n noduri și m arce, dat prin lista arcelor. Să se verifice dacă graful este hamiltonian.
5. Se consideră un graf orientat hamiltonian, cu n noduri și m arce, dat prin lista arcelor. Să se afișeze circuitele hamiltoniene ale grafului.
6. Se consideră un graf orientat cu n noduri și m arce, dat prin lista arcelor. Să se verifice dacă graful este eulerian.
7. Se consideră un graf orientat eulerian, cu n noduri și m arce, citit prin vectorul de arce. Să se afișeze toate circuitele euleriene conținute de graf.
8. Se consideră un graf orientat cu n noduri, dat prin matricea de adiacență. Să se verifice dacă graful este hamiltonian.
9. Se consideră un graf orientat hamiltonian, cu n noduri, dat prin matricea de adiacență. Să se afișeze circuitele hamiltoniene ale grafului.
10. Se consideră un graf orientat cu n noduri și m arce, dat prin matricea de adiacență. Să se verifice dacă graful este eulerian.
11. Se consideră un graf orientat eulerian cu n noduri și m arce, citit prin matricea de adiacență. Să se afișeze toate circuitele euleriene conținute de graf.
12. Numim *graf turneu*, un graf orientat în care, între oricare două noduri se află un arc și numai unul. Se citește de la tastatură un număr natural n . Să se determine un graf turneu cu n vârfuri, astfel încât între oricare două noduri să existe un drum de lungime 1 sau 2. Valoarea lui n se va citi de la tastatură. Matricea de adiacență a grafului rezultat va fi afișată în fișierul "*Turneu.txt*". În cazul în care nu există nici o soluție se va afișa un mesaj corespunzător.

13. Să se transforme un graf orientat complet în graf turneu. Graful orientat are n noduri și m arce și este citit prin lista arcelor. Să se afișeze graful turneu obținut, prin matricea de adiacență.
14. Se consideră un graf turneu cu n noduri, având arce între oricare două noduri i și j cu proprietatea că $i < j$. Să se afișeze toate drumurile elementare de la un nod x la un nod y (x și y citite de la tastatură), având proprietatea că $x < y$.
15. Să se verifice dacă un graf orientat este regulat de grad k , unde k este citit de la tastatură. Graful este dat prin matricea de adiacență (un graf este regulat dacă toate vârfurile sale au același grad).
16. Să se genereze un graf orientat complet de ordin n . Matricea de adiacență asociată grafului generat va fi afișată în fișierul "Complet.txt".
17. Fiind dată matricea de adiacență a unui graf orientat cu n noduri, se cere să se adauge un număr minim de arce, astfel încât graful să devină eulerian. Dacă acest lucru nu este posibil, se va afișa un mesaj corespunzător. Rezultatul va fi afișat într-un fișier text.

7.3. Grafuri cu costuri

1. Se dă un graf orientat în care arcele au asociate costuri (numere naturale nenule). Să se determine câte arce (x, y) din graf au costul egal cu un cost a , dat de la tastatură.
2. Se dă un graf orientat în care arcele au asociate costuri (numere naturale nenule). Să se determine câte arce (x, y) din graf au costul egal cu costul drumului de cost minim de la x la y .
3. Se consideră un graf orientat tare conex cu n noduri și m arce etichetate cu costuri pozitive. Calculați și afișați distanța maximă dintre două noduri ale grafului. Distanța de la i la j este egală cu suma costurilor arcelor ce formează drumul de cost minim de la i la j . Afișați nodurile aflate la distanță maximă unul de celălalt.
4. Să se determine într-un graf neorientat care este lungimea (numărul de muchii) lanțului minim între două vârfuri date x și y , știind că toate muchiile au același cost. (Algoritmul lui Lee)
5. Se dă lista muchiilor unui graf neorientat ponderat. Să se determine vârful pentru care media aritmetică a ponderilor muchiilor incidente este minimă. Dacă există mai multe vârfuri cu aceeași medie minimă, se va afișa vârful numerotat cu o valoare mai mică.

6. Se dă un graf orientat ponderat cu n noduri și m arce – în care fiecare arc are asociat un cost, număr natural strict pozitiv. Folosind *algoritmul Roy-Floyd*, să se construiască matricea costurilor minime.
7. Se dă un graf orientat ponderat – în care fiecare arc are asociat un cost, număr natural strict pozitiv, și un nod p . Să se determine, folosind *algoritmul lui Dijkstra*, costul minim al drumului de la p la fiecare nod al grafului.
8. Se dă un graf neorientat ponderat conex cu n vârfuri și m muchii, în care fiecare muchie are asociat un cost, număr natural strict pozitiv. Folosind *algoritmul lui Prim*, determinați un arbore parțial de cost minim, cu rădăcina în vârful 1 .
9. Se dă un graf neorientat ponderat conex cu n vârfuri și m muchii – în care fiecare muchie are asociat un cost, număr natural strict pozitiv. Folosind *algoritmul lui Kruskal*, determinați un arbore parțial de cost minim.
10. Transportul de călători între n localități ale unui județ este asigurat cu ajutorul unor microbuze. Între oricare două localități i și j există cel mult un microbuz direct, care merge o dată pe zi, având un orar cunoscut (ora de plecare din i și ora de sosire în j , sunt numere întregi). Să se găsească timpul total în care un călător poate ajunge dintr-o localitate X în altă localitate Y . Se consideră că persoana se află în stația din localitatea X la ora 0 și trebuie să ajungă în Y până la ora 24 a aceleași zile.
11. Să se realizeze o rețea de n calculatoare, de cost minim, astfel încât oricare două calculatoare să poată comunica între ele, cunoscând costul legării în rețea a oricăror două calculatoare.
12. Rețeaua de străzi a unui oraș este reprezentată printr-un graf neorientat. Vârfurile sunt de fapt intersecțiile străzilor. Străzile se consideră că au lungimi egale. Din intersecțiile x și y pleacă în același moment, două mașini care se deplasează cu viteze egale spre intersecțiile a și b . Ambele mașini se îndreaptă spre destinații pe trasee minime. Să se afișeze traseele parcurse. Să se precizeze dacă cele două mașini au trecut prin intersecții comune. În caz afirmativ să se precizeze care sunt acestea și dacă s-au întâlnit în cel puțin o intersecție. Graful este dat prin lista muchiilor.
13. Harta unui oraș este formată din n cartiere și m străzi, specificate sub forma (i, j) cu semnificația cartierul i este legat direct de cartierul j . Fiecare dintre străzi are atașat un cost, care este același, indiferent de lungimea străzii. Autoritățile orașului au stabilit costul unui abonament de autobuz ca fiind egal cu costul maxim obținut între două cartiere. Să se afișeze costul abonamentului.

7.4. Probleme olimpiadă

Problema 1 – Rover

100 puncte

NASA plănuiește o nouă misiune Rover pe Marte în anul 2020. Principalul obiectiv al acestei misiuni este de a determina, cu ajutorul unui nou Rover, dacă a existat în trecut viață pe Marte. Până când va fi lansată misiunea, Roverul este supus la tot felul de teste în laboratoarele NASA. Într-unul din teste, Roverul trebuie să parcurgă o suprafață de forma unui caroiaj cu N linii și N coloane. Acesta pornește din zona de coordonate $(1,1)$ și trebuie să ajungă în zona de coordonate (N,N) , la fiecare pas putându-se deplasa din zona în care se află într-una din zonele învecinate la nord, sud, est sau vest. Pentru fiecare zonă de coordonate (i,j) se cunoaște $A_{i,j}$, stabilitatea terenului din acea zonă. Știind că Roverul are o greutate G , o zonă cu stabilitatea terenului cel puțin egală cu G se consideră o zonă sigură pentru deplasarea Roverului, iar o zonă cu stabilitatea terenului mai mică decât G se consideră o zonă periculoasă pentru Rover.

Cerințe

1. Determinați numărul minim posibil de zone periculoase pe care le traversează Roverul pentru a ajunge din zona $(1,1)$ în zona (N,N) .
2. Determinați greutatea maximă pe care o poate avea un Rover care să ajungă din zona $(1,1)$ în zona (N,N) , fără a traversa nicio zonă periculoasă pentru el.

Date de intrare

Pe prima linie a fișierului de intrare **rover.in** se găsește numărul natural V a cărui valoare poate fi doar 1 sau 2. Dacă V este 1, pe a doua linie a fișierului de intrare se găsesc două numere naturale N și G cu semnificația din enunț, iar dacă V este 2, pe a doua linie a fișierului de intrare se află doar numărul N .

Pe următoarele N linii se află câte N numere $A_{i,j}$, reprezentând stabilitatea terenului din zona (i,j) .

Date de ieșire

Fișierul de ieșire este **rover.out**.

Dacă valoarea lui V este 1, atunci fișierul de ieșire va conține pe prima linie un număr natural reprezentând numărul minim de zone periculoase pe care trebuie să le traverseze Roverul de greutate G .

Dacă valoarea lui V este 2, atunci fișierul de ieșire va conține pe prima linie un număr natural reprezentând greutatea maximă a unui Rover care poate ajunge din zona $(1,1)$ în zona (N,N) fără a traversa zone periculoase pentru el.

Restricții și precizări

- Pentru 50% din teste $V = 1$, pentru 50 % din teste $V = 2$.
- $1 \leq N \leq 500$
- $1 \leq G \leq 5000$
- $1 \leq A_{i,j} \leq 10000$
- Zonele de coordonate $(1,1)$ și (N,N) nu sunt zone periculoase pentru Rover.
- Roverul nu va trece de mai multe ori prin aceeași zonă.

Exemplu

rover.in	rover.out	Explicație	
1 5 5 5 1 3 4 7 5 2 1 8 5 2 9 5 3 3 4 1 1 1 9 5 1 6 1 8	3	Numărul minim de zone periculoase traversate de la poziția $(1,1)$ până la poziția $(5,5)$ este 3.	Un traseu posibil este: 5 1 3 4 7 5 2 1 8 5 2 9 5 3 3 4 1 1 1 9 5 1 6 1 8
2 5 5 1 3 4 7 5 2 1 8 5 2 9 5 3 3 4 1 1 1 9 5 1 6 1 8	2	Greutatea maximă a unui Rover care poate ajunge din zona $(1,1)$ în zona $(5,5)$ fără a trece prin zone periculoase pentru el este 2.	Un traseu posibil este: 5 1 3 4 7 5 2 1 8 5 2 9 5 3 3 4 1 1 1 9 5 1 6 1 8

(Olimpiada județeană, 2017)

Problema 2– Dragoni

100 puncte

Supărați că lansarea părții a treia a filmului lor preferat s-a amânat până în iunie 2018, Henry și Hetty s-au gândit la propriul scenariu pentru finalul trilogiei:

Într-o lume în care vikingii pot zbura cu dragonii există N insule. Hiccup, șeful tribului de vikingi aflat pe insula 1, știe M rute directe de zbor **bidirecționale** între insule. Pentru fiecare j între 1 și M , ruta j unește insulele A_j și B_j și are lungime D_j .

Pe fiecare insulă i , ($1 \leq i \leq n$) există dragoni din specia i care pot zbura fără a se opri pentru odihnă o distanță maximă $D_{\max,i}$. Cu alte cuvinte, dragonii de pe insula i vor putea parcurge orice rută j , ($1 \leq j \leq m$) pentru care $D_j \leq D_{\max,i}$, indiferent de ce alte drumuri au făcut anterior.

Hiccup dorește să ajungă de pe insula 1 pe insula N pentru a-l salva pe Toothless, dragonul lui. Pentru a ajunge acolo, el va lua inițial un dragon din specia 1 (de pe insula 1). Apoi, dacă la un moment dat Hiccup se află pe o insulă i , ($1 \leq i \leq n$) având cu el un dragon din specia t , el poate:

1. Să zboare de pe insula i pe o altă insulă x cu dragonul pe care îl are, folosind o rută directă j între insulele i și x , bineînțeles doar dacă $D_j \leq D_{\max,t}$.
2. Să schimbe dragonul din specia t pe care îl are cu un dragon din specia i aflat pe insula respectivă.

Cerințe

- a. Să se determine distanța maximă $D_{\max,i}$ caracteristică unui dragon la care Hiccup poate ajunge fără a schimba dragonul pe care l-a luat inițial de pe insula 1.
- b. Să se determine distanța minimă pe care Hiccup trebuie să o parcurgă pentru a ajunge de pe insula 1 pe insula N .

Date de intrare

Fișierul de intrare `dragoni.in` conține pe prima linie un număr natural p . Pentru toate testele de intrare, numărul p poate avea doar valoarea 1 sau valoarea 2. Pe a doua linie se găsesc două numere naturale N și M reprezentând numărul de insule, respectiv numărul de rute directe între insule. Pe a treia linie se găsesc N numere naturale, al i -ulea dintre acestea reprezentând distanța maximă $D_{\max,i}$ pe care o poate zbura un dragon de pe insula i . Pe următoarele M linii sunt descrise cele M rute directe. Pe fiecare dintre aceste linii se găsesc câte trei numere naturale A , B și D cu semnificația că există rută **bidirecțională** de lungime D între insulele A și B .

Date de ieșire

În fișierul de ieșire `dragoni.out` se va afișa un singur număr natural.

Dacă valoarea lui p este 1, se rezolvă numai cerința a.

În acest caz numărul afișat va reprezenta distanța maximă $D_{\max,i}$ a unui dragon i la care Hiccup poate ajunge fără a schimba dragonul pe care l-a luat inițial de pe insula 1.

Dacă valoarea lui p este 2, se va rezolva numai cerința b,

În acest caz numărul afișat va reprezenta distanța minimă pe care Hiccup trebuie să o parcurgă pentru a ajunge de pe insula 1 pe insula N .

Restricții

- $1 \leq N \leq 800$
- $1 \leq M \leq 6000$
- $1 \leq D_{\max,i} \leq 50\,000$, pentru orice $1 \leq i \leq N$.
- $1 \leq A_j, B_j \leq N$, pentru orice $1 \leq j \leq M$.
- $1 \leq D_j \leq 50\,000$, pentru orice $1 \leq j \leq M$.
- Se garantează că Hiccup poate ajunge pe insula N .
- Se garantează că răspunsul oricărei cerințe este un număr natural mai mic decât 10^9 .
- Pentru rezolvarea corectă a primei cerințe se acordă 20% din punctajul testului respectiv.
- Pentru rezolvarea corectă a celei de-a doua cerințe se acordă 80% din punctajul testului respectiv.

Exemple

dragoni.in	dragoni.out	Explicație
1	20	$P = 1$ deci se va rezolva cerința a).
5 6		
6 3 13 20 26		Există $N = 5$ insule și $M = 6$ rute între ele.
1 2 5		Hiccup pornește de pe insula 1 având un
1 3 7		dragon care poate zbura o distanță de maxim
1 5 10		6. Cu acest dragon poate ajunge doar pe
2 3 6		insulele 1, 2, 3 și 4, întrucât pentru a ajunge
3 4 5		pe insula 5 el ar fi obligat să parcurgă o ruta
3 5 14		de lungime mai mare decât 6.
		Distanța maximă pe care o poate zbura un
		dragon aflat pe insulele 1, 2, 3 sau 4 este deci
		20 (dragonul de pe insula 4). Se observă că
		dragonul care poate zbura o distanță de 26 se
		afla pe insula 5 și este inaccesibil.

dragoni.in	dragoni.out	Explicație
2 5 6 6 3 13 20 26 1 2 5 1 3 7 1 5 10 2 3 6 3 4 5 3 5 14	28	$P = 2$ deci se va rezolva cerința b). Există $N = 5$ insule și $M = 6$ rute între ele. Pentru a parcurge o distanță minimă de 28 între insulele 1 și N, Hiccup face următorii pași: Zboară de pe insula 1 pe insula 2 o distanță de 5 cu dragonul din specia 1. Zboară de pe insula 2 pe insula 3 o distanță de 6 cu dragonul din specia 1. Schimbă dragonul din specia 1 cu dragonul aflat pe insula 3, care poate zbura o distanță maximă de 13. Zboară de pe insula 3 pe insula 1 o distanță de 7 cu dragonul din specia 3. Zboară de pe insula 1 pe insula 5 o distanță de 10 cu dragonul din specia 3. În total el parcurge o distanță de $5 + 6 + 7 + 10 = 28$.

(Olimpiada județeană, 2015)

Problema 3– Cârțițe

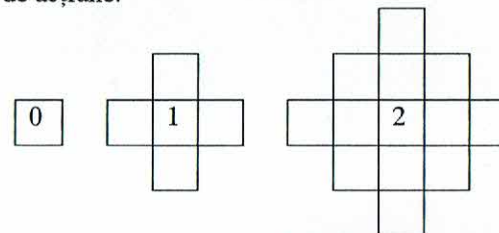
100 puncte

Cârțițele sunt animale de dimensiuni mici care își duc traiul pe suprafețe de teren deschis, având ca dușman principal vulpea. Lângă o pădure se află o zonă agricolă în formă dreptunghiulară, împărțită în pătrățele de aceeași dimensiune. Zona agricolă este reprezentată printr-un tablou bidimensional cu m linii și n coloane, având liniile și coloanele numerotate începând cu 1. În această zonă agricolă trăiește o cârțiță și k vulpi.

Pentru cârțiță cunoaștem coordonatele ei (linia și coloana) pe care se găsește, la fel și pentru vulpi, care stau la pândă pentru a ataca cârțița în momentele ei de neatenție.

Pe suprafața terenului cârțița se poate deplasa din pătrățelul în care se află doar într-unul dintre cele 4 pătrățele vecine pe direcțiile nord, sud, est sau vest.

Vulpile pot ataca instantaneu pe o raza de acțiune de lungime 0, 1 sau 2 pe orizontală și verticală, inclusiv în poziția unde se găsesc, după care tot instantaneu se întorc în pozițiile inițiale. În figura de mai jos sunt desenate pătrățelele unde poate ataca o vulpe poziționată în pătrățelul cu cifra reprezentând raza de acțiune.



Pentru a micșora riscul de deplasare în zona agricolă cârțița sapă în pământ un sistem de g galerii, care leagă între ele pătrățele din zona agricolă. Aceste galerii nu se intersectează sub pământ, ci doar la suprafață, trecerea dintr-o galerie în alta, care se intersectează în același pătrățel făcându-se printr-un sistem ce nu îi pune viața în pericol. Galerile sunt indicate prin coordonatele pătrățelelor pe care le unesc. Acestea sunt săpate astfel încât, dacă pornim dintr-un capăt al unei galerii le putem parcurge pe toate. Nu există două galerii care să pornească din același pătrățel și să ajungă tot în același pătrățel (galeriile sunt distincte).

Cârțița dorește să se plimbe prin toate galeriile de sub teren trecând o singură dată prin fiecare, dar pentru acest lucru trebuie să ajungă nevătămată mergând la suprafață terenului la un pătrățel de unde să intre în sistemul de galerii.

Cerință

Determinați:

- cel mai apropiat pătrățel de poziția inițială a cârțiței prin care ea poate să intre în galerie pentru a se plimba, precum și lungimea traseului parcurs la suprafață astfel încât fiecare pătrățică de pe traseu să nu fie atacată de nicio vulpe;

- b) traseul de plimbare numai prin galerii, specificat prin coordonatele pătrățelelor care constituie capetelor acestora.

Date de intrare

Fișierul de intrare *cartite.in* conține următoarele date:

- pe prima linie un număr natural **p**, pentru toate testele de intrare, numărul **p** poate avea doar valoarea 1 sau valoarea 2;
- pe a doua linie **m**, **n** dimensiunile zonei agricole;
- pe a treia linie coordonatele cârțiței;
- pe linia a patra numărul de vulpi **k** urmată de **k** linii cu câte trei numere naturale reprezentând coordonatele pătrățelelor unde se găsesc vulpi și raza lor de acțiune (**0**, **1** sau **2**);
- următoarea linie conține numărul de galerii **g**;
- fiecare din următoarele **g** linii conține câte **4** numere naturale separate prin câte un spațiu, reprezentând coordonatele capetelor unei galerii.

Date de ieșire

Dacă valoarea lui **p** este **1**, se va afișa numai rezultatul de la punctul a) din cerință. În acest caz, în fișierul de ieșire *cartite.out* se vor scrie trei numere naturale separate între ele prin câte un spațiu, reprezentând coordonatele pătrățelului unde va intra cârțița în galerii și lungimea traseului parcurs la suprafață.

Dacă valoarea lui **p** este **2**, se va afișa numai rezultatul de la punctul b) din cerință. În acest caz, fișierul de ieșire *cartite.out* va conține coordonatele pătrățelelor traseului de plimbare prin galerii (coordoanatele câte unui pătrățel pe câte o linie, începând cu prima linie a fișierului de ieșire). Pătrățelul de pornire trebuie să fie același cu cel în care ajunge cârțița la sfârșitul plimbării și nu este obligatoriu să fie același cu cel în care ea intră în galerii de la suprafața terenului.

Restricții și precizări

- $1 \leq m, n \leq 200$.
- $1 \leq g \leq 100$
- $0 \leq k \leq 50$
- Lungimea traseului parcurs la suprafață este egală cu numărul de pătrățele prin care aceasta trece, dar diferite de cel din care pleacă.
- Fiecare dintre cerințe reprezintă 50% din punctaj.
- Cârțița nu poate intra în galerii printr-un pătrățel din raza de acțiune a unei vulpi.
- Pentru toate testele există soluție la cerința a), adică există un traseu sigur de la cârțiță la o intrare într-o galerie.
- Soluția, nu este unică, însă, orice soluție corectă va obține punctajul maxim pe test.

- Inițial, cârțița se găsește pe o poziție în care nu este atacată de nicio vulpe.

Exemple

cartite.in	cartite.out	Explicație
1 6 4 6 3 3 5 1 0 3 4 1 4 3 0 7 1 1 3 2 1 3 1 4 1 1 3 3 1 4 4 2 4 2 3 3 4 2 1 3 4 2 3 2	4 2 3	p = 1 Deplasarea cârțiței pe suprafața agricolă se va face pe traseul de lungime 3 ce trece prin pătrățelele (6,3) (6,2) (5,2) (4,2). Coordoanatele pătrățelului de intrare în galerii sunt (4,2). Atenție! Pentru acest test se va afișa doar rezultatul la cerința a).

cartite.in	cartite.out	Explicație
2 6 4 6 3 3 5 1 0 3 4 1 4 3 0 7 1 1 3 2 1 3 1 4 1 1 3 3 1 4 4 2 4 2 3 3 4 2 1 3 4 2 3 2	1 1 3 2 4 2 1 3 1 4 4 2 3 3 1 1	p = 2 Traseul de plimbare prin galerii este următorul: (1,1) (3,2) (4,2) (1,3) (1,4) (4,2)(3,3)(1,1), cu observația că se putea alege și alt pătrățel de plecare. Atenție! Pentru acest test se va afișa doar rezultatul la cerința b). Se observă că în acest caz trebuie să se citească toate datele din fișier.

(Olimpiada județeană, 2014)

Problema 4 – Biperm

100 puncte

Pentru un număr natural nenul n , să considerăm toate numerele naturale nenule mai mici sau egale cu n , luând fiecare număr de câte două ori: $1, 1, 2, 2, 3, 3, \dots, n, n$. Aceste numere le amestecăm aleator, și le aranjăm pe două linii a câte n elemente. Structura astfel obținută o vom numi o *bipermutare*. În figurile 1, 2 și 3 avem câte un exemplu de bipermutare pentru $n=5$.

O bipermutare este *perfectă*, dacă ambele linii ale structurii reprezintă câte o permutare (vezi figurile 2 și 3).

Prin *mutare pe poziția p*, înțelegem interschimbarea elementelor de pe aceeași coloană p . În exemplele de mai jos, bipermutarea perfectă din figura 2 s-a obținut din bipermutarea din figura 1, aplicând o mutare pe poziția 2. Bipermutarea perfectă din figura 3 s-a obținut din bipermutarea din figura 1, aplicând mutări pe pozițiile 1, 2, 4 și 5.

Figura 1

Figura 2

Figura 3

Cerințe

Cunoscând o bipermutare, determinați:

- numărul bipermutarilor perfecte distincte ce se pot obține prin mutări;
- numărul minim de mutări prin care se poate obține o bipermutare perfectă;
- o bipermutare perfectă obținută din bipermutarea inițială.

Date de intrare

Fișierul de intrare `biperm.in` conține pe prima linie valoarea lui n . Următoarele două linii conțin, fiecare, câte n elemente separate prin câte un spațiu, formând o bipermutare.

Date de ieșire

Fișierul de ieșire `biperm.out` va conține:

- pe prima linie două numere naturale separate printr-un spațiu, reprezentând numărul bipermutarilor perfecte distincte ce se pot obține din bipermutarea dată, respectiv numărul minim de mutări prin care se poate obține o bipermutare perfectă;
- pe următoarele două linii se vor tipări câte n numere separate prin spațiu, reprezentând o bipermutare perfectă obținută din bipermutarea dată.

Restricții și precizări

- $2 < n \leq 10\,000$;
- calculul corect al numărului bipermutarilor perfecte distincte valorează 30% din punctaj;

• calculul corect al numărului minim de mutări valorează 10% din punctaj;

• tipărirea unei bipermutări perfecte valorează 60% din punctaj. Pot exista mai multe soluții, se va admite orice soluție corectă;

• se garantează că numărul bipermutarilor perfecte distincte nu depășește 2 000 000 000 pentru niciun test.

• acordarea punctajului la un răspuns corect este condiționată de existența răspunsurilor anterioare, indiferent de corectitudinea lor;

• pentru 40% din teste $n \leq 20$

• pentru 40% din teste $20 < n \leq 400$

• pentru 20% din teste $400 < n \leq 10\,000$

Exemplu

biperm.in	biperm.out	Explicații
5 1 5 5 3 4 3 2 2 4 1	4 1 1 2 5 3 4 3 5 2 4 1	Sunt 4 permutări perfecte. Numărul minim de mutări este 1 și există două soluții cu număr minim de mutări: 1 2 5 3 4 1 5 2 3 4 3 5 2 4 1 și 3 2 5 4 1 Celelalte două soluții, ce nu se obțin din număr minim de mutări sunt: 3 2 5 4 1 3 5 2 4 1 1 5 2 3 4 și 1 2 5 3 4 Pentru a treia cerință oricare dintre cele 4 soluții este acceptată.

(Olimpiada județeană, 2013)

8. ARBORI

8.1. Arbori cu rădăcină

1. Se consideră vectorul de tați al unui arbore cu rădăcină, cu n noduri. Determinați rădăcina arborelui și frunzele acestuia.
2. Se consideră vectorul de tați al unui arbore cu rădăcină, cu n noduri. Determinați înălțimea arborelui.
3. Se consideră vectorul de tați al unui arbore cu rădăcină cu n noduri și o valoare k . Determinați nodurile situate pe nivelul k în arbore.
4. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se afișeze nodurile care au decât descendentul stâng.
5. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se afișeze nodurile care au decât descendentul drept.
6. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se afișeze descendenții unui nod x .
7. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se afișeze câte noduri se găsesc pe nivelul p .
8. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se afișeze numărul nodurilor care au doi descendenți.
9. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se afișeze numărul de nivele ale arborelui.
10. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se afișeze suma informațiilor nodurilor terminale.
11. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se afișeze produsul informațiilor nodurilor care au decât descendent stâng.
12. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se verifice dacă toate nodurile terminale conțin ca informații, numere pare.
13. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se verifice dacă există un nod pe nivelul p a cărui informație este egală cu o valoare x dată.
14. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare naturală, să se determine maximul informațiilor conținute de noduri.
15. Fiind dat un arbore binar cu nodurile $1, 2, \dots, n$ și rădăcina r , reprezentat prin vectorii de descendenți, în care fiecare informație a nodului are o valoare întreagă, să se verifice dacă arborele este *complet*.
16. Se numește *diametru* al unui arbore distanța maximă dintre două noduri din arbore. Pentru un arbore dat, să se determine diametrul său.
17. Se citește un arbore prin vectorul de tați. Să se determine și să se afișeze cel mai lung lanț din arbore.
18. Se citește un arbore cu n vârfuri, dat prin vectorul de tați și apoi un vârf k . Să se afișeze vectorul de tați obținut prin mutarea rădăcinii arborelui în vârful k .
19. Se dau cele $n - 1$ muchii ale unui arbore cu n noduri și un nod k . Afișați vectorul de tați al arborelui cu rădăcina în k .
20. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri și k noduri distincte din arbore. Afișați fiii fiecăruia dintre cele k noduri.
21. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri și k noduri din arbore. Determinați pentru fiecare dintre cele k noduri, nivelul pe care se află.
22. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri. Determinați nodul din arbore cu număr maxim de fii. Dacă în arbore sunt mai multe noduri cu număr maxim de fii, afișați-le pe toate, în ordine crescătoare.
23. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri și un nod k . Determinați câte noduri conține subarborele cu rădăcina în k .
24. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri. Determinați câte perechi de noduri neterminale distincte p, q din arbore, au proprietatea că subarborele cu rădăcina în p și cel cu rădăcina în q au același număr de noduri.
25. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri și un nod k . Afișați, în ordine crescătoare, nodurile din subarborele cu rădăcina în k .
26. Se dă vectorul de descendenți al unui arbore cu rădăcină cu n noduri și un nod k . Afișați, în ordine crescătoare, nodurile terminale din subarborele cu rădăcina în k .
27. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri și un nod k . Determinați drumul de la nodul k la rădăcina arborelui.
28. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri și două noduri p și q . Determinați lanțul de la nodul p la nodul q .

29. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri, în care fiecare nod are asociată o valoare numerică. Determinați drumul de la rădăcină la un nod terminal, pentru care suma valorilor asociate nodurilor este maximă.
30. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri și un nod k . Fiecare nod al arborelui are asociată o valoare numerică întreagă. Determinați suma valorilor pare asociate nodurilor din subarboarele cu rădăcina în k .
31. Se dă vectorul de tați al unui arbore cu rădăcină cu n noduri. Fiecare nod al arborelui are asociată o valoare numerică întreagă. Determinați nodurile din arbore pentru care suma valorilor asociate nodurilor din subarboarele cu rădăcina în p este maximă.
32. Se dă un șir de numere naturale $d_1, d_2, \dots, d_n$. Să se verifice dacă există un arbore cu n vârfuri ale cărui grade să fie tocmai numerele date.
33. Pentru un arbore dat să se determine *codul Pruffer* asociat.
34. Se dau un număr natural n și o succesiune de $n - 2$ numere naturale, aparținând mulțimii $\{1, 2, \dots, n\}$. Să se determine un arbore al cărui *cod Pruffer* este succesiunea dată.
35. Se citește o pădure cu n vârfuri prin vectorul de tați. Să se determine din câți arbori este formată pădurea.
36. Se citește un număr natural n . Construiți un arbore cu proprietatea că fiecare vârf are numărul de descendenți direcți, cu 1 mai mare decât nivelul pe care se află. Astfel, rădăcina (aflată pe nivelul 0) are un singur descendent direct, vârful de pe nivelul 1 are 2, cele de pe nivelul 2 au 3 etc. Arborele va fi reprezentat prin vectorul de tip tată.
37. Se dau doi arbori cu rădăcină prin doi vectori de tip tată. Determinați dacă cei doi arbori cu rădăcină reprezintă același arbore (diferă doar în urma alegerii rădăcinilor).
38. Se citește un arbore cu n vârfuri dat prin vectorul muchiilor și vârful rădăcină. Să se calculeze și să se afișeze numărul de niveluri ale arborelui.
39. Se dă un arbore cu n noduri ($1 \leq n \leq 100000$), fiecare nod având un cost ($0 \leq \text{cost} \leq 10000$). Se cere să determinați *centroidul* acestui arbore. *Centroidul* unui arbore este nodul pentru care, dacă este ales ca rădăcină, numărul maxim de noduri dintr-unul din subarborii determinați de fiii săi, este minim.
40. Se dă un arbore cu n ($1 \leq n \leq 100000$) noduri, fiecare nod având un cost ($0 \leq \text{cost} \leq 10000$). Să se găsească o submulțime de noduri de cost minim, astfel încât fiecare muchie din arbore să aibă incident cel puțin un nod din submulțime. Costul unei submulțimi se definește ca fiind suma costurilor fiecărui nod din submulțime.
Datele de intrare se citesc din fișierul de intrare "*Submultime.in*", care conține pe prima linie un număr întreg n , reprezentând numărul de noduri.

A doua linie va conține n valori, reprezentând costurile nodurilor $1, 2 \dots n$. Următoarele $n - 1$ linii vor conține fiecare câte 2 valori, x, y cu semnificația că există muchie (neorientată) de la x la y . Programul va afișa costul minim al unei submulțimi astfel încât orice muchie să fie incidentă la cel puțin un nod din acea submulțime.

41. Se consideră un arbore cu n ($1 \leq n \leq 10000$) vârfuri, în care fiecare vârf i are asociată o valoare întreagă c_i ($-1000 \leq c_i \leq 1000$). Să se determine un subarbore al arborelui dat, pentru care suma valorilor asociate vârfurilor conținute în subarbore să fie maximă.

Datele de intrare se citesc din fișierul de intrare "*Maxime.in*", care conține pe prima linie numărul întreg n , reprezentând numărul de vârfuri ale arborelui. A doua linie a fișierului conține n valori întregi, reprezentând valorile asociate nodurilor. A i -a valoare din acest șir reprezintă valoarea asociată nodului i . Următoarele $n - 1$ linii conțin câte doi întregi distincți a și b , separați printr-un spațiu, având semnificația că există muchie între vârfurile a și b . Programul va afișa suma maximă a unui subarbore al arborelui dat.

8.2. Arbori binari alocați dinamic. Arbore parțial de cost minim

1. Să se creeze un arbore binar cu informații numere întregi, apoi să se determine cel mai mare număr din arbore.
2. Să se creeze un arbore binar cu informații numere întregi, apoi să se determine suma cifrelor tuturor numerelor din arbore.
3. Să se creeze un arbore binar cu informații numere întregi. Să se calculeze produsul cheilor pozitive ale arborelui creat.
4. Să se creeze un arbore binar cu informații numere întregi. Să se afișeze cheile impare din arbore.
5. Să se creeze un arbore binar cu informații numere întregi. Să se afișeze nodurile din arbore care au exact un succesor.
6. Să se construiască un arbore binar și apoi să se caute o valoare dată, specificându-se dacă este în arbore.
7. Se dă un arbore binar cu informații numere întregi. Să se afișeze frunzele arborelui.
8. Să se determine, pentru o arborescență dată, numărul de niveluri ale acesteia.
9. Fiind dat un arbore binar, să se determine pentru fiecare vârf nivelul său.

10. Să se determine, pentru un arbore binar dat, nivelul maxim al acestuia (adâncimea arborelui).
11. Să se scrie un program pentru listarea tuturor nodurilor aflate pe un nivel dat, într-un arbore binar.
12. Se dă un arbore binar conținând numere naturale nenule. Să se determine valoarea maximă memorată în nodurile sale.
13. Se dă un arbore binar cu informații numere naturale. Să se verifice dacă arborele conține numere prime.
14. Să se creeze un arbore binar cu informații numere întregi, apoi să se tipărească suma elementelor impare din arbore.
15. Să se scrie un program care afișează toți descendenții din dreapta ai rădăcinii.
16. Să se scrie un program care verifică dacă un arbore binar este arbore de căutare.
17. Să se scrie o funcție care returnează numărul de nivele ale unui arbore binar (înălțimea arborelui). Funcția va primi ca parametru un pointer p către rădăcina arborelui.
18. Să se scrie un subprogram care tipărește informațiile din nodurile aflate pe un nivel dat. Subprogramul va primi ca parametri numărul nivelului, precum și un pointer către rădăcina arborelui.
19. Să se afle înălțimea unui arbore cu rădăcină, adică distanța maximă de la frunze la rădăcină.
20. Să se afle diametrul unui arbore, adică distanța maximă între oricare două frunze.
21. Se dă un arbore binar cu cheile numere întregi. Să se scrie un program care va înlocui toate aparițiile nodului cu cheia a din arbore, cu valoarea b (a și b sunt date).
22. Să se parcurgă un arbore în *preordine*, *inordine* și *postordine*.
23. Să se scrie un program care afișează cel de-al p -lea element obținut în urma traversării în *inordine* a unui arbore binar.
24. Să se scrie un program care afișează cel de-al p -lea element obținut în urma traversării în *preordine* a unui arbore binar.
25. Să se scrie un program care afișează cel de-al p -lea element obținut în urma traversării în *postordine* a unui arbore binar.
26. Să se scrie un program care, pe baza succesiunii nodurilor în *inordine* și *preordine* ale unui arbore binar, determină succesiunea în *postordine* a acestora.

27. Să se scrie un algoritm care, pe baza succesiunilor nodurilor în *inordine* și *postordine* ale unui arbore binar, determină succesiunea în *preordine* a acestora.
28. Fiind dat un arbore oarecare, să se determine arborele binar asociat lui.
29. Se consideră un arbore binar de căutare. Să se scrie un program care să realizeze linializarea acestuia, adică să copieze informațiile din noduri într-un vector.
30. Fiind date un arbore binar și două vârfuri ale sale, de pe niveluri diferite, să se decidă dacă unul dintre ele este descendent al celuilalt.
31. Să se determine o modalitate de colorare a vârfurilor unui arbore binar dat, folosind două culori, astfel încât oricare două vârfuri adiacente să fie colorate diferit. Culorile vor fi codificate prin litere. Rezultatul va fi trecut într-un fișier text.
32. Să se construiască un arbore binar cu n vârfuri date, având adâncimea minimă.
33. Să se scrie un program care să verifice dacă un arbore binar este simetric (coincide cu imaginea sa în oglindă).
34. Să se afișeze informațiile atașate nodurilor unui arbore binar, utilizând parantezele rotunde pentru a marca nodurile situate pe același nivel în arbore, ca în exemplul 1 (2 (5 (. , 6) , .) , 3 (. , 4 (7 , .))))
35. Se dă o expresie aritmetică de forma $a_1 + a_2 + \dots + a_m$, unde $a_1, a_2, \dots, a_m$ sunt litere. Se cere construirea arborelui binar asociat acestei expresii, dacă evaluarea ei s-ar face astfel: $a_1 + (a_2 + (a_3 + \dots + a_m))$.
36. Se dă o expresie aritmetică de forma $a_1 + a_2 + \dots + a_m$, unde $a_1, a_2, \dots, a_m$ sunt litere. Se cere construirea *arborelui binar AVL* asociat și apoi notația poloneză prefixată.
37. Să se realizeze un program de gestiune a unei librării, folosind un arbore binar (nodurile arborelui vor conține următoarele informații: cod, nume autor, titlu și editura). Vor fi permise operațiile: căutarea unei cărți după cod, adăugarea unei noi cărți, listarea tuturor cărților și ștergerea unei cărți, după un cod dat.

8.3. Probleme olimpiadă

Problema 1 - Ninjago

După ce eroii ninja l-au învins pe Nadakhan, de ziua celor dispăruți Zane trebuia să păzească cele n păpuși din muzeu. Între aceste păpuși există coridoare pe care se poate circula în ambele sensuri. Se garantează faptul că pe cele m coridoare Zane poate ajunge la fiecare dintre cele n păpuși. Skulkiu, având la dispoziție 5 tipuri de obstacole A, B, C, D, E, încearcă să-l oprească pe Zane punând pe fiecare coridor câte 4 obstacole. Zane poate distruge obstacolele de tip A, B, C și D, dar nu poate să distrugă obstacolele de tipul E. Pentru a distruge un obstacol de tipul Arma lui Zane are nevoie de 1 unitate de energie, pentru a distruge un obstacol de tipul B de 2 unități de energie, pentru a distruge un obstacol de tipul C de 3 unități de energie, iar pentru a distruge un obstacol de tipul D de 4 unități de energie. Datorită dispozitivului cu care Skulkiu amplasează obstacolele pe coridor, cele patru obstacole de pe același coridor au o adâncime din ce în ce mai mare, ceea ce implică faptul că pentru a distruge al doilea obstacol amplasat pe coridor este nevoie de 5 ori mai multă energie decât cea obișnuită, pentru a distruge cel de-al treilea obstacol amplasat pe coridor este nevoie de 25 ori mai multă energie decât cea obișnuită, iar pentru a distruge al patrulea obstacol amplasat pe același coridor este nevoie de 125 de ori mai multă energie decât cea obișnuită. Indiferent de sensul de parcurgere al coridorului de către Zane pentru a înlătura obstacolele, energia consumată este aceeași, aceasta depinzând doar de ordinea în care au fost amplasate obstacolele de către Skulkiu. Zane nu va înlătura obstacolele de pe toate coridoarele ci doar strictul necesar pentru a avea acces la fiecare păpușă. Zane dorește să-l lase pe ceilalți ninja să se antreneze așa că face în așa fel încât ajutorul pentru distrugerea obstacolelor de tip E să fie minim și apoi ca el să utilizeze un număr minim de unități de energie. Pentru coridoarele pe care se află obstacole de tip E Zane consumă energie doar pentru obstacolele de tip A, B, C și D. Inițial Zane se află lângă păpușa 1.

Cerințe

- 1) Precizați la câte dintre cele n păpuși poate ajunge Zane înainte de a cere ajutorul celorlalți ninja.
- 2) Precizați pentru eliberarea câtor coridoare trebuie să ceară ajutor extern pentru a reuși să ajungă la toate cele n păpuși și câte obstacole de tip E sunt în total pe aceste coridoare.
- 3) Precizați care este numărul minim de unități de energie utilizate.

Date de intrare

Fișierul **ninjago.in** conține pe prima linie un număr natural v care poate avea doar valorile 1, 2 sau 3 reprezentând cerința care va fi rezolvată. Fișierul **ninjago.in** conține pe a doua linie numerele naturale n și m separate printr-un spațiu, iar pe următoarele m linii pentru fiecare coridor două numere naturale separate printr-un spațiu reprezentând cele două păpuși între care se circulă pe

coridorul respectiv urmate de un spațiu și patru litere corespunzătoare celor patru tipuri de obstacole în ordinea în care Skulkiu le-a amplasat pe coridorul respectiv. Între cele patru litere nu se află nici un spațiu.

Date de ieșire

Dacă valoarea lui v este 1 atunci fișierul de ieșire **ninjago.out** va conține pe prima linie numai numărul păpușilor la care poate ajunge Zane înainte de a cere ajutorul celorlalți ninja.

Dacă valoarea lui v este 2 atunci fișierul de ieșire **ninjago.out** va conține pe prima linie numărul de coridoare pe care nu le poate elibera singur și pe a doua linie numărul total de obstacole de tip E de pe aceste coridoare.

Dacă valoarea lui v este 3 atunci fișierul de ieșire **ninjago.out** va conține pe prima linie numai numărul minim de unități de energie utilizate.

Restricții și precizări

- $1 \leq N, M \leq 31200$;
- Pentru rezolvarea corectă a primei cerințe se acordă 20 de puncte;
- Pentru rezolvarea corectă a celei de-a doua cerințe se acordă 40 de puncte;
- Pentru rezolvarea corectă a celei de-a treia cerințe se acordă 30 de puncte.
- Problema va fi evaluată pe teste în valoare de 90 de puncte.
- Se vor acorda 10 puncte din oficiu.

Exemple

ninjago.in	ninjago.out	explicații
1	5	Zane va putea ajunge la nodurile 1, 2, 5, 6, 7
915		eliberând singur coridoarele
1 2 CBAA		(1,2),(1,5),(2,7) și (6,7)
1 5 ABAA		
2 6 CBEA		
2 7 ACBA		
2 5 CBEA		
3 4 ABAA		
3 7 AECE		
3 8 CBAC		
3 9 ECEE		
4 8 DBAD		
4 9 ECEB		
5 6 CBAD		
5 7 BAAD		
6 7 CBAA		
7 8 ECEB		

ninjago.in	ninjago.out
2	2
9 15	4
1 2 CBAA	
1 5 ABAA	
2 6 CBEA	
2 7 ACBA	
2 5 CBEA	
3 4 ABAA	
3 7 AECE	
3 8 CBAC	
3 9 ECEE	
4 8 DBAD	
4 9 ECEB	
5 6 CBAD	
5 7 BAAD	
6 7 CBAA	
7 8 ECEB	

ninjago.in	ninjago.out
3	1593
9 15	
1 2 CBAA	
1 5 ABAA	
2 6 CBEA	
2 7 ACBA	
2 5 CBEA	
3 4 ABAA	
3 7 AECE	
3 8 CBAC	
3 9 ECEE	
4 8 DBAD	
4 9 ECEB	
5 6 CBAD	
5 7 BAAD	
6 7 CBAA	
7 8 ECEB	

observații

Zane trebuie să ceară ajutor pentru eliberarea a 2 coridoare pentru a putea ajunge la toate cele 9 păpuși
Zane are nevoie de ajutor pentru a elibera coridoarele (3,7) și (4,9). Pe fiecare dintre aceste 2 coridoare se află câte 2 obstacole de tip E, deci în total 4 obstacole de tip E.

observații

Zane va consuma minim 1593 de unități de energie astfel:

163 pentru coridorul (1,2)
161 pentru coridorul (1,5)
191 pentru coridorul (2,7)
161 pentru coridorul (3,4)
76 pentru coridorul (3,7)
413 pentru coridorul (3,8)
265 pentru coridorul (4,9)
163 pentru coridorul (6,7)

Pentru coridorul (4,9) obstacolele sunt ECEB,
Deci Zane va consuma
 $0+3*5+0*25+2*125=265$ unități de energie

(Olimpiada județeană, 2017)

9. STRUCTURI DE DATE

9.1. Stiva

- Să se scrie un program care gestionează o stivă de numere întregi. Inițial stiva este vidă. Programul va citi de la tastatură o listă de operații, care pot fi:
 - *push* – adaugă valoarea întreagă x pe stivă;
 - *pop* – elimină elementul din vârful stivei;
 - *top* – afișează elementul din vârful stivei.
 Afișarea va fi făcută într-un fișier text.
- Să se scrie funcția de adăugare pentru o stivă alocată ca vector.
- Să se scrie funcția de extragere a unui element dintr-o stivă alocată ca vector.
- Să se scrie funcția de concatenare a două stive alocate dinamic S_1 și S_2 , folosind doar operațiile de bază (adăugare, extragere, testare stivă vidă). Rezultatul trebuie să conțină elementele din cele două stive în ordinea inițială.
- Se consideră un șir de numere întregi. Să se scrie funcția care construiește două stive (una cu numerele negative și una cu cele pozitive) ce conțin numerele în ordinea inițială folosind doar structuri de tip stivă.
- Se dau n șiruri de paranteze rotunde. Să se stabilească, despre fiecare șir, dacă este corect parantezat.
- Se dau n șiruri de paranteze rotunde sau pătrate. Să se stabilească, despre fiecare șir, dacă este corect parantezat.
- Se dau un șir de paranteze rotunde care se închid corect (corect parantezat). Să se determine cât de adâncă este parantezarea (numărul de paranteze deschise/închise).
- Se consideră un cuvânt format numai din litere mici și un număr natural nenul k . Să se determine cuvântul minim lexicografic obținut prin eliminarea a exact k litere din cuvântul inițial.
- Se consideră o expresie corectă formată din numere naturale și operatorii $+$, $-$, $*$. Să se evalueze expresia, ținându-se cont de operatorii precizați.
- Se citește din fișier un șir de numere întregi pozitive, terminat cu -1 . Să se sorteze acest șir prin inserție directă, cu ajutorul a două stive. În prima stivă se rețin numerele ordonate (descrescător, astfel încât la baza stivei se află cel mai mare). La citirea unui nou număr din fișier, x , se extrag din această stivă toate numerele mai mici decât cel citit, salvându-le în a doua stivă. Inserăm numărul x în stiva ordonată și apoi reintroducem numerele salvate (temporar) din a doua stivă.
- Se citește dintr-un fișier o expresie aritmetică în care operanzii sunt simbolizați prin litere mici (de la a la z) iar operatorii sunt $+$, $-$, $*$, $/$, $($, $)$, cu semnificația cunoscută. Expresia se încheie cu caracterul $."$ (caracterul punct). Se cere să se convertească această expresie într-o expresie în forma poloneză (postfixată).

13. Să se scrie funcția de inversare a unui șir folosind o stivă.
14. Scrieți funcția recursivă pentru ștergerea tuturor elementelor dintr-o stivă.
15. Scrieți un program care citește de la tastatură un șir de caractere și verifică dacă șirul este palindrom. Programul va utiliza o stivă pentru realizarea cerinței.

9.2. Coadă

1. Să se scrie un program care gestionează o coadă de numere întregi. Inițial coada este vidă. Programul va citi de la tastatură o listă de operații, care pot fi:
 - *push* – adaugă valoarea întregă x în coadă;
 - *pop* – elimină elementul din coadă;
 - *front* – afișează elementul de la începutul cozii.
 Afișarea se va face într-un fișier text.
2. Se introduc de la tastatură n valori într-o coadă alocată ca vector. Să se scrie funcția de adăugare a unui element.
3. Se introduc de la tastatură n valori într-o coadă alocată ca vector. Să se scrie funcția de ștergere a unui element.
4. Un grup de n copii se așează în cerc. Numerotarea se face în sensul acelor de ceas cu $1, 2, \dots, n$. În jocul lor, ei folosesc o frază de tipul “ala bala portocala ...” (fraza se citește dintr-un fișier text). Fiecăruia, începând cu primul, i se asociază un cuvânt din frază. Cel care primește ultimul cuvânt este eliminat din cerc. Jocul continuă, începând fraza de la următorul copil, până când se elimină toți copiii. Folosind numerotarea inițială, să se afișeze ordinea ieșirii copiilor din joc. (*Problema lui Josephus*).
5. Se citesc de la tastatură elementele unei cozi. Să se scrie un program care creează o nouă coadă care conține toate elementele de pe poziții pare din prima coadă.
6. Să se memoreze n numere întregi într-o structură de tip coadă. Să se șteargă elementele neprime din coadă, până se întâlnește un număr prim.
7. Scrieți funcția pentru ștergerea tuturor elementelor dintr-o coadă alocată dinamic.

10. LISTE ALOCATE DINAMIC

10.1. Liste simplu înlănțuite

10.1.1. Parcurgeri

1. Să se scrie un program care afișează pe ecran valorile memorate într-o listă simplu înlănțuită.
2. Să se scrie un program care determină câte elemente negative sunt memorate într-o listă simplu înlănțuită.
3. Să se creeze o listă simplu înlănțuită. Să se afișeze elementele numere pare conținute de ea.
4. Să se creeze o listă simplu înlănțuită. Să se calculeze suma elementelor de pe poziții impare din lista dată.
5. Să se creeze o listă simplu înlănțuită. Să se afișeze elementele numere prime din listă.
6. Să se creeze o listă simplu înlănțuită. Să se afișeze elementele numere perfecte din listă.
7. Să se creeze o listă simplu înlănțuită. Să se verifice dacă o valoare dată x se găsește în listă. În caz afirmativ, să se afișeze numărul de apariții.
8. Se consideră o listă simplu înlănțuită. Să se determine câte dintre elementele listei se află în intervalul $[a, b]$ (a, b sunt date).
9. Se dă o listă simplu înlănțuită. Să se afișeze elementele din listă care au suma cifrelor mai mică decât un număr k dat.
10. Se dă o listă liniară simplu înlănțuită. Să se afișeze elementele consecutive din listă care au aceeași paritate.
11. Să se scrie un program care determină câte perechi de elemente consecutive, prime între ele sunt memorate într-o listă simplu înlănțuită.
12. Să se scrie un program care determină câte perechi de elemente consecutive egale sunt memorate într-o listă simplu înlănțuită.
13. Se consideră o listă liniară simplu înlănțuită cu elemente numere întregi. Să se afișeze toate perechile de elemente consecutive din listă, cu proprietatea că ambele elemente ale perechii au același număr de cifre.
14. Se dă o listă liniară simplu înlănțuită, cu elemente numere întregi. Să se calculeze media aritmetică a elementelor pozitive.
15. Se dă o listă liniară simplu înlănțuită, cu elemente numere întregi. Să se calculeze media aritmetică pentru oricare două elemente consecutive din listă.
16. Se consideră o listă liniară simplu înlănțuită, cu elemente numere întregi. Să se determine câte dintre elementele din listă sunt strict mai mici decât ma și câte sunt strict mai mari decât ma , unde ma este media aritmetică a elementelor din listă.

17. Se dă o listă liniară simplu înălțuită cu elemente numere întregi. Să se determine diferența în valoarea absolută dintre numărul de valori pozitive și numărul de valori negative.
18. Se dă o listă liniară simplu înălțuită cu elemente numere întregi. Să se determine câte perechi de numere consecutive din listă au produsul un număr par.
19. Se dă o listă liniară simplu înălțuită cu elemente numere întregi. Să se determine câte perechi de numere consecutive din listă au produsul un număr negativ.
20. Să se scrie un program care determină suma elementelor impare dintr-o listă simplu înălțuită, care sunt situate între două elemente pare.
21. Să se scrie un program care sortează crescător elementele unei liste simplu înălțuite.
22. Să se scrie un program care sortează decrescător elementele unei liste simplu înălțuite începând de la poziția k din listă (k este dat).
23. Să se creeze o listă liniară simplu înălțuită cu cifrele unui număr natural dat de la tastatură. Cifrele se vor introduce în listă începând cu cifra unităților.
24. Să se scrie un program care verifică dacă două liste liniare simplu înălțuite sunt identice.
25. Să se verifice dacă o listă simplu înălțuită formează un palindrom.
26. Să se creeze o listă liniară simplu înălțuită cu primele n numere prime.
27. Se consideră o listă liniară simplu înălțuită cu elemente numere naturale. Să se afișeze elementele din listă care sunt prime și care citite invers sunt tot numere prime.
28. Să se creeze o listă cu primii n termeni din *șirul lui Fibonacci*.
29. Să se determine câți termeni din *șirul lui Fibonacci* conține o listă liniară simplu înălțuită.
30. Să se formeze o listă liniară care conține elementele unui vector. Cele n elemente ale vectorului sunt date de la tastatură.
31. Să se formeze o listă liniară care conține elementele pozitive ale unui vector. Cele n elemente ale vectorului sunt date de la tastatură.
32. Să se formeze o listă liniară care conține elementele numere prime ale unui vector. Cele n elemente ale vectorului sunt date de la tastatură.
33. Să se formeze o listă care conține elementele unei matrici. Elementele matricii sunt date de la tastatură. Vor fi introduse în listă pe linii.
34. Să se formeze o listă care conține elementele numere perfecte ale unei matrici. Elementele matricii sunt date de la tastatură. Vor fi introduse în listă pe linii.

35. Fișierul "*Date.in*" conține pe primul rând numere întregi. Să se construiască o listă liniară simplu înălțuită care să conțină numerele din fișier.
36. Fișierul "*Date.in*" conține pe primul rând numere întregi. Să se construiască o listă liniară simplu înălțuită care să conțină numerele pozitive din fișier.
37. Fișierul "*Date.txt*" conține pe primul rând numere întregi. Să se creeze și să se afișeze o listă liniară simplu înălțuită care va conține valorile numere prime din fișierul dat.
38. Fișierul "*Date.txt*" conține pe primul rând numere întregi. Să se creeze și să se afișeze o listă liniară simplu înălțuită care va conține valorile numere perfecte din fișierul dat.
39. Se consideră o listă liniară simplu înălțuită cu n elemente numere întregi. Să se inverseze sensul legăturilor din listă astfel încât să se obțină o listă cu nodurile aranjate în ordine inversă.
40. Să se scrie un program care verifică dacă două liste date sunt disjuncte.
41. Să se afișeze elementele (elementul) din mijlocul unei liste cu numere întregi.
42. Să se creeze o listă care prelucrează numere complexe. Lista se citește din fișier, pe fiecare linie fiind scrise două valori reprezentând partea reală și respectiv cea imaginară. Să se calculeze suma numerelor complexe date și să se afișeze rezultatul tot într-un fișier text.
43. Să se creeze o listă care prelucrează numere complexe. Lista se citește din fișier, pe fiecare linie fiind scrise două valori reprezentând partea reală și respectiv cea imaginară. Să se determine numărul complex care are modulul cel mai mare. Rezultatul va fi afișat într-un fișier text.
44. Să se sorteze crescător elementele numere pare ale unei liste simplu înălțuite cu n elemente numere întregi.
45. Se consideră fișierul "*Date.in*" care conține n numere întregi dispuse pe prima linie. Să se determine un subșir de lungime maximă, cu elementele ordonate crescător (ordinea apariției elementelor din fișier se păstrează și în subșir) și să se memoreze subșirul astfel obținut într-o listă liniară.
46. Se consideră două polinoame $P(x)$ și $Q(x)$ de grade n și m , reprezentate prin liste liniare simplu înălțuite. Fiecare nod va conține un coeficient al polinomului și puterea corespunzătoare a lui x . Să se calculeze $P(x) + Q(x)$ și $P(x) \cdot Q(x)$. Rezultatele vor fi trecute în câte o listă liniară simplu înălțuită și vor fi afișate într-un fișier text.

10.1.2. Construcție liste

1. Se consideră o listă liniară simplu înlănțuită cu elemente numere întregi. Să se construiască o listă care să conțină elementele pozitive ale primei liste.
2. Se consideră o listă liniară simplu înlănțuită cu elemente numere întregi. Să se construiască o listă care să conțină elementele de pe pozițiile pare ale primei liste.
3. Se consideră o listă liniară simplu înlănțuită cu elemente numere întregi. Să se construiască o listă care să conțină elementele pare de pe poziții impare ale primei liste.
4. Se consideră o listă liniară simplu înlănțuită cu elemente numere întregi. Să se construiască o listă care să conțină elementele numere prime ale primei liste.
5. Se consideră o listă liniară simplu înlănțuită cu elemente numere întregi. Să se construiască o listă care să conțină elementele care au suma cifrelor egală cu o valoare k (k este dat de la tastatură).
6. Se dă o listă cu numere naturale. Să se creeze două liste cu elementele listei inițiale, astfel încât una va conține numerele pare, iar cea de a doua numerele impare. Să afișeze listele astfel obținute.
7. Se consideră o listă liniară simplu înlănțuită cu elemente numere întregi. Să se construiască o listă care să conțină elementele din lista inițială, aflate în intervalul $[a, b]$ (a și b sunt date).
8. Se dă o listă simplu înlănțuită cu n elemente numere întregi. Să se creeze o a doua listă care să conțină elementele numere prime din prima listă.
9. Se consideră o listă simplu înlănțuită L , cu elemente numere întregi. Să se construiască o listă T astfel încât T_i să conțină numărul de apariții ale lui L_i în L .
10. Se consideră o listă liniară simplu înlănțuită cu elemente numere întregi. Să se creeze o a doua listă care să conțină elementele din lista inițială, împreună cu frecvențele lor de apariție.
11. Fișierul "*Multimi.txt*" conține pe primele două rânduri două șiruri de numere, reprezentând două mulțimi A și B . Elementele celor două mulțimi sunt memorate în 2 liste liniare simplu înlănțuite. Să se creeze o a treia listă liniară simplu înlănțuită, care să memoreze reuniunea celor 2 mulțimi.
12. Fișierul "*Multimi.txt*" conține pe primele două rânduri două șiruri de numere, reprezentând două mulțimi A și B . Elementele celor două mulțimi sunt memorate în 2 liste liniare simplu înlănțuite. Să se creeze o a treia listă liniară simplu înlănțuită care să memoreze intersecția celor 2 mulțimi.
13. Fișierul "*Multimi.txt*" conține pe primele două rânduri două șiruri de numere, reprezentând două mulțimi A și B . Elementele celor două mulțimi sunt memorate în 2 liste liniare simplu înlănțuite. Să se creeze o a treia listă liniară simplu înlănțuită care să memoreze diferența celor 2 mulțimi.

10.1.3. Operații cu liste

1. Să se scrie un program care adaugă o valoare x la finalul unei liste simplu înlănțuite.
2. Să se scrie un program care adaugă o valoare x la începutul unei liste simplu înlănțuite.
3. Să se scrie un program care adaugă o valoare x pe poziția k într-o listă simplu înlănțuită, cu n elemente numere întregi. Valorile lui k și x sunt citite de la tastatură.
4. Să se scrie un program care adaugă o valoare y după valoarea x , dintr-o listă simplu înlănțuită cu n elemente numere întregi. Valoarea lui x este citită de la tastatură. În cazul în care există mai multe valori x , valoarea y se va adăuga după fiecare apariție a lui x .
5. Să se scrie un program care adaugă o valoare înainte de valoarea x , dintr-o listă simplu înlănțuită cu n elemente numere întregi. Valoarea lui x este citită de la tastatură. În cazul în care există mai multe valori x , valoarea y se va adăuga înainte de fiecare apariție a lui x .
6. Se consideră o listă simplu înlănțuită. Să se scrie un program care adaugă după fiecare element de pe pozițiile impare, un număr de elemente egal cu valoarea elementului de pe poziția respectivă.
7. Se consideră o listă simplu înlănțuită cu n elemente numere întregi. Să se adauge după fiecare element, suma cifrelor sale.
8. Se consideră o listă simplu înlănțuită cu n elemente numere întregi. Să se adauge după fiecare element, media aritmetică a elementelor pozitive din listă.
9. Să se creeze o listă liniară simplu înlănțuită care conține elemente numere întregi, citite dintr-un fișier text. Se citesc două valori x și y . Să se adauge după primul nod care conține valoarea x un nod care să conțină valoarea y .
10. Se consideră o listă liniară simplu înlănțuită, cu elemente numere întregi. Să se grupeze zerourile listei la sfârșitul ei (Se presupune că lista conține cel puțin trei zerouri).
11. Să se scrie un program care șterge primul element al unei liste simplu înlănțuite.
12. Să se scrie un program pentru ștergerea ultimului element dintr-o listă simplu înlănțuită.
13. Să se scrie un program care șterge elementul de pe poziția k dintr-o listă liniară simplu înlănțuită.
14. Să se scrie un program care șterge dintr-o listă simplu înlănțuită un element x dat. În cazul în care lista are mai multe elemente x , se vor șterge toate.
15. Se consideră o listă liniară simplu înlănțuită cu n elemente numere întregi. Să se șteargă elementul aflat după un element x dat, din listă. Elementul x poate apărea de mai multe ori în listă.

16. Se consideră o listă liniară simplu înălțuită cu n elemente numere întregi. Să se ștergă elementul aflat înaintea unui element x dat, din listă. Elementul x poate apărea de mai multe ori în listă.
17. Se consideră o listă simplu înălțuită cu n elemente numere întregi. Să se elimine din listă elementele x_i pentru care suma elementelor alăturate este egală cu x_i .
18. Se consideră o listă simplu înălțuită cu n elemente numere întregi. Să se elimine din listă elementele x_i , pentru care ma a elementelor $x_1, x_2, \dots, x_{i-1}$ este x_i .
19. Să se scrie un program care șterge dintr-o listă simplu înălțuită toate elementele pare.
20. Să se scrie un program pentru ștergerea tuturor elementelor numere prime dintr-o listă simplu înălțuită. Se presupune că lista are cel puțin un element număr prim.
21. Se se scrie un program pentru ștergerea tuturor elementelor dintr-o listă simplu înălțuită care au suma cifrelor divizibilă cu o valoare x dată.
22. Se consideră o listă liniară simplu înălțuită cu n elemente numere întregi. Să se ștergă din listă toate elementele pare care sunt încadrate de 2 elemente numere impare.
23. Se consideră un șir de n numere întregi, cu n număr natural nenul. Se elimină primul element din șir și toate elementele șirului aflate pe poziții care reprezintă numere prime, în ordinea crescătoare a pozițiilor. Operația de eliminare se repetă cu elementele rămase în șir, repositionate după eliminarea celorlalte, până când este eliminat și ultimul element rămas. Să se scrie un program care a fișează elementele șirului inițial, în ordinea în care au fost eliminate conform algoritmului descris mai sus. (Admitere FMI București – 2016)
24. Să se scrie un program care inserează după fiecare element par al unei liste simplu înălțuită, dublul său.
25. Să se scrie un program care inserează înaintea de fiecare element impar al unei liste simplu înălțuită, dublul său.
26. Fie o listă simplu înălțuită de numere întregi distincte. Să se insereze primele două valori (cele mai mari) la începutul listei.
27. Să se scrie un program care inserează înaintea fiecărui element par al unei liste simplu înălțuită, suma cifrelor sale.
28. Se consideră o listă liniară simplu înălțuită. Să se scrie un program care mută ultimul nod din listă în fața primului.
29. Se consideră o listă liniară simplu înălțuită. Să se scrie un program care mută nodul de pe poziția k , pe prima poziție în listă.
30. Să se scrie un program care inserează într-o listă simplu înălțuită cu elementele ordonate crescător, un nou nod care memorează valoarea x astfel încât informațiile din listă să fie în continuare ordonate crescător.
31. Se consideră o listă liniară simplu înălțuită. Să se insereze între oricare două elemente consecutive, suma lor.

32. Să se creeze o listă simplu înălțuită cu numerele naturale preluate dintr-un fișier. Numerele sunt dispuse pe mai multe linii ale fișierului de intrare "Numere.in", cele de pe aceeași linie fiind despărțite de câte un spațiu. Se cere să se adauge după fiecare număr din listă, răsturnatul său. Să se afișeze conținutul listei înainte și după modificare, în fișierul "Numere.out", fiecare afișare pe câte un rând.
33. Să se creeze o listă simplu înălțuită cu numerele întregi preluate dintr-un fișier. Numerele sunt dispuse pe mai multe linii ale fișierului "Cifre.in", cele de pe aceeași linie fiind despărțite de câte un spațiu. Se cere să se adauge după fiecare număr din listă, cifra de control a numărului respectiv. Să se afișeze conținutul listei înainte și după modificare în fișierul "Cifre.out", fiecare afișare pe câte un rând.
34. Să se creeze o listă liniară simplu înălțuită care conține elemente numere reale citite dintr-un fișier text "Lista.txt". Să se insereze între oricare două noduri din listă un nod care să conțină media aritmetică a celor 2 valori din cele 2 noduri.
35. Se consideră o listă liniară simplu înălțuită. Să se scrie un program care inserează după fiecare element de pe o poziție pară x , un număr de x elemente egale cu elementele de pe acea poziție.
36. Să se permute o listă liniară simplu înălțuită cu k poziții (se vor permuta decât informațiile nodurilor).
37. Se consideră o listă liniară simplu înălțuită cu n elemente numere întregi. Să se scrie un program care interschimbă primul nod cu al doilea, al treilea cu al patrulea ș.a.m.d.
38. Se consideră două liste liniare simplu înălțuite cu elemente numere întregi și cu elementele dispuse în ordine crescătoare. Să se concateneze cele două liste, astfel încât lista rezultată să fie ordonată crescător.

10.2. Liste circulare

1. Se citește de la tastatură un vector cu n elemente numere întregi. Să se construiască o listă circulară, care să conțină elementele vectorului.
2. Să se adauge un element la sfârșitul unei liste circulare. Lista astfel obținută va fi tot circulară.
3. Să se ștergă elementul de pe ultima poziție a unei liste circulare. Lista astfel obținută va fi tot circulară.
4. Se consideră două liste circulare, cu elementele dispuse în ordine crescătoare. Să se concateneze cele două liste, astfel încât elementele listei finale să fie dispuse tot crescător.
5. Se consideră o listă simplu înălțuită. Să se determine dacă lista este liniară sau circulară.
6. Se consideră o listă circulară simplu înălțuită, cu elemente numere întregi. Se presupune că lista conține cel puțin trei zerouri. Să se grupeze zerourile listei la sfârșitul ei.

7. Se consideră o listă circulară simplu înlănțuită, cu elemente numere întregi. Se presupune că lista conține cel puțin 2 numere prime. Să se grupeze numerele prime ale listei la sfârșitul ei.
8. Să se afișeze elementele unei liste circulare începând cu o poziție p dată.
9. Să se permute o listă circulară simplu înlănțuită cu k poziții și să se afișeze.
10. Să se permute o listă circulară, cu câte o poziție spre dreapta, până când lista ajunge la lista inițială. Listele rezultate vor fi afișate în fișierul "Permutari.txt", câte una pe un rând.
11. Se consideră o listă circulară simplu înlănțuită cu elemente numere întregi, citite de la tastatură. Să se parcurgă cele n elemente pornind de la primul, din p în p elemente. Valoarea lui p este dată de la tastatură.
12. Se așează un număr de n copii în cerc. Numele celor n copii se găsesc în fișierul "Joc.txt". Copiii s-au hotărât să joace un joc, folosindu-se de silabele unui cântec. Fișierul "Joc.txt" conține silabele cântecelului despărțite prin câte o liniuță. De la tastatură se citește numele unui copil de la care se începe "numărătorea". Numele copilului în dreptul căruia se va rosti ultima silabă va ieși din cerc. Să se determine ordinea copiilor care ies din cerc.

10.3. Liste dublu înlănțuite

1. Se consideră o listă dublu înlănțuită cu n elemente numere întregi. Să se afișeze elementele din listă de la stânga spre dreapta și de la dreapta spre stânga.
2. Se consideră o listă dublu înlănțuită cu n elemente numere întregi. Să se afișeze elementele pozitive din listă de la stânga spre dreapta și de la dreapta spre stânga.
3. Se consideră o listă dublu înlănțuită cu n elemente numere întregi. Să se afișeze elementele prime din listă în ordinea inversă introducerii lor în listă.
4. Se consideră o listă liniară dublu înlănțuită cu număr par de noduri. Să se scrie un program care interschimbă primul nod cu al doilea, al treilea cu al patrulea ș.a.m.d.
5. Se consideră o listă liniară dublu înlănțuită. Să se șteargă primul, ultimul și nodul de pe poziția k din listă.
6. Se consideră o listă liniară dublu înlănțuită. Să se scrie o funcție care primește ca parametru adresa primului nod al listei și mută ultimul nod în fața primului.
7. Se consideră o listă liniară dublu înlănțuită în ale cărei noduri sunt memorate cifre. Să se scrie o funcție care primește ca parametru adresa primului nod al listei și verifică dacă numărul care se compune din cifrele memorate este palindrom.
8. Să se creeze o listă liniară dublu înlănțuită care să memoreze valori întregi. Să se scrie o funcție care primește ca parametru adresa primului nod al listei și mută primul nod după ultimul. Folosind această funcție, să se afișeze permutările circulare ale șirului de numere memorat în listă.

9. Să se transforme o listă circulară simplu înlănțuită, într-o listă liniară dublu înlănțuită.
10. Să se scrie un program pentru adunarea a două matrice rare, memorate ca liste dublu înlănțuite.
11. Să se scrie un program pentru calculul sumei a două numere mari. Cifrele numerelor mari vor fi memorate în nodurile a două liste liniare dublu înlănțuite.
12. Să se scrie un subprogram recursiv care afișează elementele unei liste liniare dublu înlănțuite, în ambele sensuri, în funcție de opțiune.
13. Se consideră două liste liniare dublu înlănțuite cu elementele în ordine crescătoare. Să se concateneze cele două liste, astfel încât elementele din lista rezultată să fie ordonate tot crescător.
14. Să se scrie un program care concatenează n liste dublu înlănțuite ordonate crescător.
15. Să se construiască o listă liniară dublu înlănțuită cu n elemente numere întregi. Să se elimine din listă primul și ultimul element egal cu un număr x dat.
16. Să se construiască o listă liniară dublu înlănțuită cu n elemente numere întregi. Să se elimine din listă elementele care se repetă. Lista astfel rezultată, va fi afișată într-un fișier text.
17. Se consideră o listă liniară dublu înlănțuită cu n elemente numere întregi. Să se adauge în fața unui element x , media aritmetică a elementelor dinaintea lui și după el, suma elementele pozitive, aflate după el în lista inițială.
18. Se consideră o listă liniară dublu înlănțuită în ale cărei noduri sunt memorate cifre. Să se scrie o funcție care primește ca parametru adresa primului nod al listei și verifică dacă numărul format din cifrele memorate în listă, în ordinea în care apar, este sau nu palindrom.
19. Să se memoreze într-o listă dublu înlănțuită n numere întregi. Să se interschimbe elementele simetrice: primul element cu ultimul, al doilea cu penultimul etc.
20. Să se șteargă prima și ultima valoare pozitivă dintr-o listă liniară dublu înlănțuită.
21. Se consideră o listă liniară dublu înlănțuită. Să se insereze între oricare două elemente consecutive, suma lor.
22. Se consideră o listă liniară dublu înlănțuită cu elemente numere întregi. Să se mute numerele pozitive la sfârșitul listei. Nu se va folosi o listă intermediară.
23. Să se determine cea mai lungă secvență crescătoare dintr-o listă dublu înlănțuită.
24. Se consideră o listă liniară dublu înlănțuită cu elemente numere întregi. Să se șteargă din listă, cea mai lungă secvență crescătoare.
25. Se consideră o listă liniară dublu înlănțuită cu elemente numere întregi. Să se afișeze perechile de numere simetrice din listă (primul cu ultimul, al doilea cu penultimul etc.). Rezultatul va fi trecut în fișierul text "Perechi.txt", fiecare pereche pe câte un rând din fișier.

BIBLIOGRAFIE

1. E. Cerchez – Culegere de probleme pentru liceu, Editura Polirom, Iași, 2002;
2. D. Hrinciuc Logofătu – C++. Probleme rezolvate și algoritmi, Editura Polirom, Iași, 2001;
3. C. Ionescu, G. Bălan – Informatică pentru grupele de performanță, Clasa a X-a, Editura Dacia Educațional, Cluj-Napoca, 2004;
4. C. Ionescu, M. Scorțaru – Informatică pentru grupele de performanță, Clasa a XI-a, Editura Dacia Educațional, Cluj-Napoca, 2004;
5. C. Ivașcu, M. Prună – Tehnici de programare, Editura Petrion, București, 1999
6. D. Lica, M. Pașoi – Fundamentele programării clasa a IX-a, Editura L&S Infomat, București, 2018;
7. D. Lica, M. Pașoi – Fundamentele programării clasa a X-a, Editura L&S Infomat, București, 2018;
8. D. Lica, M. Pașoi – Fundamentele programării clasa a XI-a, Editura L&S Infomat, București, 2018;
9. V. Lungu – Programare C/C++, Editura Printech, București, 2002
10. G. Mateescu, P.F. Moraru – Informatica pentru liceu și Bacalaureat – Materia de clasa a IX-a, Editura Donaris, Sibiu, 2001;
11. G. Mateescu, P.F. Moraru – Informatica pentru liceu și Bacalaureat – Materia de clasa a XI-a, Editura Donaris, Sibiu, 2001;
12. G. Mateescu, P.F. Moraru – Informatica pentru liceu și Bacalaureat – Materia de clasa a X-a, Editura Donaris, Sibiu, 2001;
13. C. Mincă – Informatică. Caiet de laborator pentru clasa a X-a, Editura L&S Infomat, București, 2018
14. C. Mincă – Informatică. Caiet de laborator pentru clasa a XI-a, Editura L&S Infomat, București, 2018
15. MEC – Subiecte propuse la Examenul de Bacalaureat în anii 2009-2018
16. V. Mitrană – Provocarea algoritmilor. Probleme pentru concursurile de informatică, Editura Agni, 1994
17. C. Popescu – Culegere de probleme de informatică, Editura Donaris-Info, Sibiu, 2002
18. D. Popescu Anastasiu – Culegere de probleme de informatică pentru gimnaziu și liceu, Editura All Educațional, București, 2000
19. M. Runceanu, C. Negrea – Culegere de probleme de informatică, Editura Donaris-Info, Sibiu, 2003
20. L.D. Satinoianu – Informatică clasa a XI-a – Fișe de lucru pentru elevi, Editura Donaris-Info, Sibiu, 2002
21. T. Sorin – Manual pentru clasa a X-a, Editura L&S Infomat, București, 2012
22. T. Sorin, V. Huțanu – Manual pentru clasa a XI-a, Editura L&S Infomat, București, 2012
23. C. Udrea – Pascal, teorie și aplicații, clasa a X-a, Editura Arves, Craiova, 2002
24. C. Udrea – Pascal, teorie și aplicații, clasa a XI-a, Editura Arves, Craiova, 2002

Site-uri

1. <http://info.mcip.ro/>
2. <https://www.pbinfo.ro/>
3. <http://elf.cs.pub.ro>
4. <http://runceanu.ro>
5. <https://invata.info>