

**Barem de notare pentru toate problemele:**

**Oficiu:**

- 1 punct (din oficiu)
- 1 punct (fără erori de compilare)
- 1 punct (definirea tuturor variabilelor)
- 2 puncte (citirea corectă a datelor de intrare, validarea acestora în funcție de enunț)
- 4 puncte (implementarea unui algoritm corect: prime, cifre, sortare, prelucrare caractere...)
- 1 punct (afișarea rezultatelor)

## Vectori:

1. Se citesc 2 numere naturale  $a$  și  $b$  de la tastatură ( $a < b$ ), de maxim 9 cifre fiecare. Să se scrie în fișierul *fibonacci.txt*, separate prin câte un spațiu toate numerele din șirul lui Fibonacci care aparțin intervalului  $[a, b]$ .

**Exemplu: Dacă se citește  $a=4$ ,  $b=30$**

**Conținutul fișierului fibonacci.txt va fi: 5 8 13 21**

2. Fișierul text *numere.txt* conține pe o singură linie, separate prin câte un spațiu, cel mult 100 de numere naturale, fiecare număr având cel mult 4 cifre. Scrieți un program care citește numerele din fișierul *numere.txt* și afișează pe ecran, separate prin câte un spațiu, în ordine crescătoare, toate numerele naturale nenule din fișier. Dacă nu există astfel de numere se va afișa pe ecran mesajul **NU EXISTA**.

**Exemplu: Dacă conținutul fișierului numere.txt este: 567 312 9 0 400**

**Pe ecran se va tipări: 9 312 400 567**

3. Se citește de la tastatură un număr natural  $n$  ( $n \geq 100$ ). Să se determine dacă numărul dat este număr „munte-vale”. Un număr natural este „munte-vale” dacă cifrele din număr sunt în ordine crescătoare până la o anumită poziție, iar cifrele care urmează sunt în ordine descrescătoare până la sfârșit.

**Exemple pentru numere „munte-vale”: 24521, 18942, 16432.**

4. Se citesc de la tastatură două tablouri unidimensionale cu elementele în ordine crescătoare, **a** cu **n** elemente, respectiv **b** cu **m** elemente. Să se determine din cele două tablouri, un al treilea tablou unidimensional în mod optim, care să conțină toate elementele celor două tablouri tot în ordine crescătoare. Tabloul nou format va fi afișat în fișierul *ordonat.txt*.

**Dacă datele de intrare sunt:**

**4**

**1 2 3 4**

**5**

**1 3 5 7 9**

**Conținutul fișierului ordonat.txt va fi: 1 1 2 3 3 4 5 7 9**

5. Se citesc de la tastatură două tablouri unidimensionale cu elementele în ordine crescătoare, **a** cu **n** elemente, respectiv **b** cu **m** elemente. Să se realizeze dacă sunt mulțimi, reuniunea și diferența lor. În cazul în care un tablou nu reprezintă o mulțime cu elemente distincte, se va afișa mesajul **”X-nu e multime”**, unde X – este tabloul **a** sau **b**. Rezultatul va fi scris în fișierul *multime.txt*.

**Exemplu: 4**

**1 3 5 6**

**5**

**1 3 5 7 9**

**Multime.txt: 1 3 5 6 7 9 (reuniunea)**

**6 (diferenta)**

6. Fișierul text **interval.txt** conține pe prima linie un număr natural nenul  $n$  ( $1 \leq n \leq 1000$ ), iar pe fiecare dintre următoarele  $n$  linii câte două numere întregi  $a$  și  $b$  ( $1 \leq a \leq b \leq 32000$ ), fiecare pereche reprezentând un interval închis de forma  $[a, b]$ . Scrieți un program care citește numerele din fișier și determină un interval dintre cele citite care conține cel mai mare număr de numere întregi și afișează pe o linie a ecranului, separate printr-un spațiu, numerele care reprezintă capetele intervalului determinat. În cazul în care sunt mai multe intervale care îndeplinesc această proprietate, se vor afișa informațiile referitoare la acel interval la care numărul care reprezintă capătul din dreapta este minim.

Exemplu:

```
4
17 24
-2 3
9 15
8 15
```

Se va afișa **8 15**.  $[8, 15]$  și  $[17, 24]$  au același număr de elemente întregi, dar 8 e mai mic.

7. Fișierul text **cifre.txt** conține pe prima linie un număr natural  $n$  ( $0 < n < 1000$ ), iar pe a doua linie  $n$  numere naturale cu cel mult 9 cifre fiecare. Scrieți un program care citește toate numerele din fișier și afișează pe ecran, separate prin câte un spațiu, numerele formate doar din cifre distincte și care au exact trei cifre.

**Exemplu: dacă fișierul cifre.txt are următorul conținut:**

```
7
249 511 4329 2 4313 243 3562
atunci pe ecran se vor afișa numerele 249 243.
```

8. Din fișierul **vector.in** se citește un vector de numere întregi, pozitive, cu cel puțin 2 cifre fiecare. Sa se determine cea mai lungă subsecvență de elemente prime, ale căror inverse sunt tot numere prime.

Exemplu: dacă fișierul **vector.in** are următorul conținut:

```
9
11 971 44 19 181 751 347 33 929
atunci subsecvența cerută este: 181 751 347
```

9. Se dă un vector de numere întregi, pozitive, care se citește din fișierul **vector.in**. Se cere să se afișeze subsecvența palindromică de lungime maximă.

Exemplu: dacă fișierul **vector.in** are următorul conținut:

```
15
1 12 31 12 1 4 27 13 9 26 9 13 27 4 131
atunci subsecvența cerută este: 4 27 13 9 26 9 13 27 4
```

10. Se citește de la tastatură un număr natural  $n$  și elementele unui tablou unidimensional cu  $n$  elemente numere întregi între 1 și 10, apoi să se determine un tablou unidimensional, astfel încât pe fiecare poziție  $i$  să avem numărul de apariții a numărului din tabloul inițial. Rezultatul trebuie scris într-un fișier *aparitii.out*.

Exemplu: Pentru  $n=9$ ,  $v=(1, 5, 2, 1, 5, 7, 2, 1, 5)$  se obține  $w=(3, 3, 2, 3, 3, 1, 2, 3, 3)$ .

## Matrici:

- De la tastatură se citește un număr natural  $n$  ( $n \leq 10$ ) apoi un vector cu  $n \cdot n$  elemente numere întregi. Să se completeze matricea circular cu elementele vectorului în ordinea citirii. Matricea formată va fi afișată în fișierul *matrice.txt*.

**Exemplu:**  $n=4$ , vectorul este: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16.

**Matricea afișată va fi:**

```

1 2 3 4
12 13 14 5
11 16 15 6
10 9 8 7

```

- Din fișierul *vecini.txt* se citește un număr  $n$ , urmat de o matrice pătratică cu elemente numere întregi. Să se afișeze pe linii diferite ale ecranului coordonatele elementelor, separate prin câte un spațiu, care au ca vecini (8 direcții) numai numere impare.

**Exemplu:**  $n=4$ , iar elementele din matrice:

```

1      1      3      4
5      6      7      8
9      9      11     12
13     14     15     16

```

**Rezultatul afișat va fi:** 2 2

- Scrieți programul care citește de la tastatură un număr natural  $n$  ( $n \leq 50$ ) și construiește în memorie o matrice cu  $n$  linii și  $n$  coloane, ale cărei elemente sunt numere întregi citite de la tastatură. Pentru fiecare coloană a matricei, în ordine, programul afișează în fișierul *minim.txt* cel mai mic număr de pe respectiva coloană. Numerele afișate vor fi separate prin câte un spațiu.

$n=4$

```

1      12     13     45
5      16     7      18
9      10     11     12
13     14     15     16

```

**Conținutul fișierului minim.txt va fi:** 1 10 7 12

4. Fie fișierul **maxim.txt** care conține pe prima linia două numere naturale  $n$  și  $m$ . Începând cu linia a doua a fișierului găsim cele  $n*m$  elemente ale unei matrici. Să se determine prima valoare maximă din matrice, care se va afișa pe ecran alături de coordonatele sale. Să se elimine din matrice linia și coloana unde s-a găsit această valoare maximă. Matricea va fi afișată după ștergere.

Conținutul fișierului **maxim.txt** va fi:

4 5

1	12	13	45	10
5	16	7	18	49
9	50	11	12	25
13	14	15	16	34

Valorile afișate vor fi:

50 3 2

1	13	45	10
5	7	18	49
13	15	16	34

5. Din fișierul **matrice.in** se citește de pe primul rând două valori  $n$  și  $m$  ce reprezintă numărul de linii și de coloane ale unei matrici și apoi  $n*m$  valori întregi. Să se rotească matricea cu 90 de grade în sens trigonometric și să se afișeze noua matrice.

Dacă conținutul fișierului **maxim.txt** este:

4 5

1	12	13	45	10
5	16	7	18	49
9	50	11	12	25
13	14	15	16	34

Valorile afișate vor fi:

10	49	25	34
45	18	12	16
13	7	11	15
12	16	50	14
1	5	9	13

6. Se citește de la tastatură un număr natural  $n$  – impar. Să se genereze în fișierul **matrice.txt** o matrice de forma:

N=3	N=5	N=7
1 1 1	1 1 1 1 1	1 1 1 1 1 1 1
1 3 1	1 3 3 3 1	1 3 3 3 3 3 1
1 1 1	1 3 5 3 1	1 3 5 5 5 3 1
	1 3 3 3 1	1 3 5 7 5 3 1
	1 1 1 1 1	1 3 5 5 5 3 1
		1 3 3 3 3 3 1
		1 1 1 1 1 1 1

7. Scrieți un program care citește de la tastatură un număr natural nenul. Fie  $x$ - numărul de cifre al numărului citit. Construiți în memorie și afișați apoi pe ecran o matrice având  $x$  linii și  $x$  coloane, completată astfel: elementele de pe coloană  $x$  a matricei vor fi toate egale cu cifra unităților numărului dat, elementele de pe coloana  $x-1$  a matricei vor fi toate egale cu cifra zecilor... Prima coloană va conține prima cifră a numărului dat.

N=123	N=10038
1 2 3	1 0 0 3 8
1 2 3	1 0 0 3 8
1 2 3	1 0 0 3 8
	1 0 0 3 8
	1 0 0 3 8

8. Din fișierul **matrice.in** se citește de pe primul rând două valori  $n$  și  $m$  ce reprezintă numărul de linii și de coloane ale unei matrici și apoi  $n*m$  valori întregi. Să se afișeze toate elementele de tip „șă”. Numim elemente de tip „șă” acele elemente care sunt maxime pe linia de care aparțin și minime pe coloană sau invers. Pentru fiecare element de tip „șă” se va tipări linia, coloana și valoarea elementului.

**Exemplu:**

**n=4 m=3**

**1 6 3**

**5 8 2**

**4 7 2**

**5 8 1**

Se va tipări: 1 2 6 (explicație: elementul  $a[1][2]=6$  este maxim pe linia 1 și minim pe coloana 2)

9. Fișierul **matrice.in** conține pe primul rând două valori  $n$  și  $m$  ce reprezintă numărul de linii și de coloane ale unei matrici și apoi  $n*m$  valori întregi. Să se determine cea mai mare valoare aflată pe rama matricii și de câte ori apare această valoare.

Numim *ramă a matricii* cadrul format din prima linie, ultima coloană, ultima linie și prima coloană.

Conținutul fișierului **matrice.in** va fi:

4 5

10            49            25            34

49            18            12            16

13            7            11            15

12            16            50            14

1            5            9            49

Valorile afișate vor fi:

49 3

10. Se dă o matrice pătratică  $A$  cu  $n$  linii și  $n$  coloane, cu valori pozitive. Să se scrie un program care citește matricea, înlocuiește fiecare valoare de pe diagonalele principală și secundară cu suma elementelor de pe linia respectivă și apoi afișează matricea într-un fișier text **rez.txt** (pe fiecare rând, o linie a matricii).

Ex:  $n=4$

3	6	1	2		12	6	1	12
1	9	4	7		1	21	21	7
4	2	6	3	Devine:	4	15	15	3
2	5	1	8		16	5	1	16



## Șiruri de caractere:

- Din fișierul **cuvinte.txt** se citesc mai multe cuvinte, fiecare de maxim 20 caractere, scrise pe linii diferite. Să se afișeze aceste cuvinte în ordine crescătoare, ținând cont de numărul de vocale din fiecare.

Exemplu:

<i>cuvinte.txt</i>	<i>Pe ecran</i>
Atestat informatica Bac programator Pascal Cpp Competente	Cpp Bac Pascal Atestat Competente programator informatica

- Scrieți un program care citește de la tastatură un cuvânt format din cel mult 20 de litere mici ale alfabetului englez și afișează pe ecran, toate “clonele” (*s2 este “clona” șirului de caractere s1 dacă se poate obține din s1 prin eliminarea tuturor aparițiilor unei singure vocale*) acestui cuvânt, fiecare pe câte o linie a ecranului.

Exemplu: pentru cuvântul **informatica** se afișează, nu neapărat în această ordine, “clonele” scrise alăturat: **nformatca, infrmatica, informtic**.

- Scrieți un program care citește de la tastatură două caractere c1 și c2 și un text având cel mult 250 caractere (spații și litere ale alfabetului englez), pe care îl modifică înlocuind toate aparițiile caracterului memorat în c1 cu cel memorat în c2 și toate aparițiile caracterului memorat în c2 cu cel memorat în c1. Programul afișează pe linii separate ale ecranului atât textul inițial cât și textul obținut după efectuarea înlocuirilor.

**Exemplu: dacă c1=a și c2=e, iar textul este: “ana are mere”**

**Se va tipări:**

**ana are mere**

**ene era mara**

- Se citește un text de la tastatură. Să se insereze după fiecare vocală mică, vocala mare corespunzătoare. Noul șir format se va scrie în fișierul **vocale.txt**.

Ex. ”informatica” va deveni ”**iInfoOrmaAtiIcaA**”.

5. Din fișierul *sufix\_prefix.txt* se citesc 2 texte de maxim 50 caractere situate pe linii diferite. Pentru primul cuvânt să se afișeze toate sufixele sale pornind de la 1 caracter. Pentru cel de al doilea text să se afișeze prefixele sale pornind de la șirul inițial.

**Exemplu:** în fișierul de intrare avem "mate", "info", atunci pe ecran vom obține:

```
e
te
ate
mate
info
inf
in
i
```

6. Un șir cu maximum 255 de caractere conține cuvinte separate prin unul sau mai multe spații. Cuvintele sunt formate numai din litere mici ale alfabetului englez. Scrieți un program care citește un astfel de șir și îl afișează în fișierul *litere.txt* modificat, prima și ultima literă a fiecărui cuvânt fiind afișată ca literă mare.

**Ex:** "atestat la informatica" va fi "AtestaT LA InformaticA".

7. Din fișierul *text.in* se citește o valoare naturală  $n$  și apoi de pe următoarele  $n$  rânduri câte un cuvânt. Să se numere și să se afișeze câte dintre cuvintele citite încep și se încheie cu o vocală.

**Exemplu:**

```
6
ana
are
mere
scumpe
si
ulei
```

Se va tipări: 3 (este vorba de cuvintele **ana**, **are**, **ulei**)

8. Din fișierul *text.in* se citește o valoare naturală  $n$  și apoi de pe următoarele  $n$  rânduri câte un cuvânt. Să se numere și să se afișeze câte dintre cuvintele citite au număr egal de vocale și de consoane.

**Exempu:**

```
6
ana
are
mere
scumpe
si
ulei
```

Se va tipări: 2(este vorba de cuvintele **mere**, **si**)

9. Fiind dat un șir de maximum 100 de caractere care conține cuvinte formate din literele alfabetului englez, separate prin câte un spațiu. Să se genereze un nou șir în care vocalele să se multiplice de un număr egal de ori cu pozițiile lor din șirul inițial.

**Exemplu:**

**Șirul dat:** Cerul este alb.

**Șirul rezultat:** Ceeruuuul eeeeeesteeeeeeeeee aaaaaaaaaaaaalb.

10. Fie fișierul text *in.txt*, care conține un text de maximum 200 de caractere. Să se scrie un program care codifică textul dat în felul următor: după fiecare vocală inserează litera “p” și vocala. Textul codificat va fi afișat pe ecran.

**Exemplu:** dacă conținutul fișierului *in.txt* este „Mere pere banana”,

**atunci pe ecran se va afișa:** „Meperepe peperepe bapanapanapa”

11. Se citește un șir de caractere de forma: cifră-literă, cifră literă ...etc. Să se construiască și să se tipărească în fișierul *sir.out* decodificarea acestui șir, în care fiecare literă apare de atâtea ori cât este cifra care o precede.

**Exemplu:** pentru șirul de intrare 2a4b5c, în fișierul de ieșire *sir.out* se va afișa aabbbbccccc