

Probleme pentru ATESTAT - programare | 2014

Probleme propuse pentru examenul de ATESTAT la INFORMATICĂ Secțiunea PROGRAMARE

Nr. crt.	Problema
1.	Problema 1
2.	Problema 2
3.	Problema 3
4.	Problema 4
5.	Problema 5
6.	Problema 6
7.	Problema 7
8.	Problema 9
9.	Problema 10
10.	Problema 11
11.	Problema 12
12.	Problema 13
13.	Problema 20
14.	Problema 21
15.	Problema 22
16.	Problema 23
17.	Problema 24
18.	Problema 25
19.	Problema 26
20.	Problema 27

Probleme pentru ATESTAT - programare | 2014

1. Din fișierul text *vector.in* se citesc: un număr natural n reprezentând numărul de componente ale unui vector cu numere naturale și cele n componente ale vectorului ($n < 50000$). Fiecare componentă va fi un număr natural cu cel puțin două cifre!
 - a) Afișați elementele vectorului pe primul rând al fișierului text *vector.out*, separate prin câte un spațiu.
 - b) Sortați crescător vectorul, după ultimele două cifre ale fiecăruia din cele n numere ce reprezintă componentele vectorului, apoi afișați, pe al doilea rând al fișierului *vector.out* componentele vectorului sortat, separate prin câte un spațiu.

Exemplu: Dacă fișierul *vector.in* are forma:

```
5
122 13 1445 9000 321
atunci fișierul vector.out va conține:
122 13 1445 9000 321
9000 13 321 122 1445
```

2. Din fișierul text *vector.in* se citesc: un număr natural n ($n < 10000$) reprezentând numărul de componente ale unui vector cu numere naturale și cele n componente ale vectorului.
 - a) Afișați componentele vectorului, pe primul rând al fișierului text *vector.out*, în ordine inversă citirii lor.
 - b) Pe al doilea rând al fișierului text *vector.out* afișați media aritmetică a componentelor vectorului, care sunt divizibile cu 3. Există cel puțin un număr multiplu de 3.

Exemplu: Dacă fișierul *vector.in* are forma:

```
5
12 6 48 28 33
atunci fișierul vector.out va conține:
33 28 48 6 12
24.75
```

3. Din fișierul text *vector.in* se citesc: un număr natural n ($n < 2000$) și n perechi de numere a , b sunt două numere întregi ce reprezintă capetele unor intervale închise la ambele capete, de forma $[a, b]$ ($a < b$).
Pe primul rând al fișierului text *vector.out*, afișați suma obținută prin adunarea tuturor numerelor întregi aflate în intervalul obținut prin intersectarea celor n intervale date. Dacă intersecția este mulțimea vidă, atunci se va afișa mesajul „Niciun element!”.

Exemplu: Dacă fișierul *vector.in* are forma:

```
3
-3 2
-2 10
-2 7
atunci fișierul vector.out va conține:0
```

Explicație: Intervalul obținut prin intersectarea celor n intervale este $[-2, 2]$.

4. Este sfârșitul școlii. Dirigintele clasei dorește să premieze elevii cei mai buni. Pentru aceasta are nevoie de o ierarhizare a acestora, în ordinea descrescătoare a mediilor. Știind că, pentru fiecare din cei n elevi ai clasei se cunosc numele și media generală, ajutați-l pe diriginte să facă premiarea.

Datele se citesc din fișierul *vector.in* astfel:

- Pe prima linie se află numărul n de elevi;
- Pe următoarele n linii se află numele elevului și media, separate prin câte un spațiu.

În fișierul de ieșire *vector.out* se vor afișa pe n linii, separate prin câte un spațiu, numele elevului și media, în ordinea descrescătoare a mediilor.

Exemplu: Dacă fișierul *vector.in* are forma:

```
3
Popoviciu 7.86
Ionovici 9.47
Arhivescu 8.88
```

atunci fișierul *vector.out* va conține:

```
Ianovici 9.47
Arhivescu 8.88
Popoviciu 7.86
```

5. Din fișierul *vector.in* se citesc: n un număr natural nenul reprezentând numărul de componente ale unui vector cu numere naturale nenule și cele n componente ale vectorului.

- a) Afișați, pe primul rând al fișierului *vector.out* mesajul „Exista” sau „Nu exista”, dacă printre componentele vectorului există vreun număr perfect. Un număr este perfect dacă este egal cu suma divizorilor săi.
- b) În cazul în care, există cel puțin un număr perfect, afișați, pe al doilea rând al fișierului *vector.out* suma tuturor numerelor perfecte din șirul dat. Dacă nu există niciun număr cu proprietatea cerută, pe al doilea rând al fișierului de ieșire se va afișa -1.

Exemplu: Dacă fișierul *vector.in* are forma:

```
6
13 33 6 28 6 16
```

atunci fișierul *vector.out* va conține:

```
Exista 40
```

Explicație: Numărul 6 este perfect deoarece $6=1+2+3$

6. Din fișierul text *date.in* se citesc numere întregi. Să se determine cel mai mare divizor propriu al fiecărui număr citit. Afișarea va fi făcută în fișierul *date.out*.

Exemplu :

date.in
6 8 15 21

date.out
Divizorul maxim al elementului 6 este 3
Divizorul maxim al elementului 8 este 4
Divizorul maxim al elementului 15 este 5
Divizorul maxim al elementului 21 este 7

7. Din fișierul *date.in* se citesc de pe prima linie un număr natural n , iar de pe următoarele n linii o matrice patratică de dimensiune $n \times n$. Să se afișeze suma elementelor de pe fiecare linie. Rezultatul va fi afișat în fișierul *date.out* sub forma:

```
linie 1 ->suma_1
linie 2 ->suma_2
.....
```

linie n -> suma_n

Exemplu :

date.in

4
2 5 3 6
3 5 8 4
1 2 6 3
8 3 5 9

date.out

linie 1 -> 16
linie 2 -> 20
linie 3 -> 12
linie 4 -> 25

9. Din fișierul *contur.in* se citește n , număr natural ($2 \leq n \leq 20$) și o matrice pătratică de dimensiune $n \times n$. Realizați un program care afișează în fișierul *contur.out* :

a) conturul matricei

b) câte elemente prime se află pe conturul matricei

Exemplu: Pentru $n=3$ și matricea:

1 2 3
4 5 6
7 8 9

Se va afișa:

1 2 3 6 9 8 7 4
3

10. Realizați un program care determină cel mai mare divizor comun al elementelor situate pe fiecare coloană a unei matrice pătratice.

Exemplu: pentru $n=3$ și matricea

4 12 36
3 15 90
7 6 30

se va afișa 4, 3, 1.

11. Se citește un șir de n (număr natural $1 < n \leq 10$) numere naturale. Se formează un număr cu ultima cifră a fiecărui număr din șir. Să se scrie un program care afișează pe ecran dacă numărul astfel obținut este palindrom (adică dacă citit de la stânga la dreapta este egal cu cel citit de la dreapta la stânga).

Exemplu: Pentru $n=6$ și numerele 11 22 43 151 17 3, se va afișa mesajul "nu este palindrom".

12. Să se citească din fișiereul text „*date.in*” un șir de maximum 255 de caractere format din cuvinte separate prin unul sau mai multe spații. Cuvintele sunt formate numai din litere mici ale alfabetului englez. Scrieți un program C/C++ care citește un astfel de șir și afișează pe ecran frecvența de apariție a fiecărei litere din șir.

Exemplu:

Pentru șirul: competente profesionale

Se va afișa:

a apare de 1 ori

c apare de 1 ori

e apare de 5 ori
f apare de 1 ori
i apare de 1 ori
l apare de 1 ori
m apare de 1 ori
n apare de 2 ori
o apare de 3 ori
p apare de 2 ori
r apare de 1 ori
s apare de 1 ori
t apare de 2 ori

13. Să se citească de la tastatură două șiruri formate din maximum 25 de caractere fiecare. Fiecare șir conține numai litere mici ale alfabetului englez. Scrieți un program C/C++ care verifică și afișează printr-un mesaj sugestiv, dacă cele două șiruri sunt sau nu anagrame (să conțină aceleași litere, în orice ordine de apariție).

Exemplu:

Pentru șirurile „curat” și „urcat” se afișează “sirurile sunt anagrame”

Pentru șirurile „problema” și „emblema” se afișează “sirurile nu sunt anagrame”

20. Se citește de la tastatură o serie de n numere întregi. Afișați primul și ultimul număr par din serie. Afișările se vor face cu text semnificativ, câte o informație pe linie. Dacă nu există cel puțin 2 numere pare în serie se va afișa textul “Nu exista”.

Example:

1. $n = 6$, seria : 6 1 2 7 4 9, se va afișa 6 și 4.

2. $n = 4$, seria : 6 1 3 7 5 9, se va afișa “Nu exista”.

21. În fișierul text “atestat.in” se găsesc pe prima linie două numere întregi n și k ($1 < k < n \leq 1000$), iar pe următoarea linie n numere întregi separate prin spații. Afișați în fișierul text “atestat.out” primele k minime din seria de n numere de pe linia a doua a fișierului de intrare.

Exemplu: $n=5$, $k=3$, seria este: 3, 2, -3, 6 și -1, se va afișa -3, -1, 2.

22. Se citesc de la tastatură două numere întregi n și m . Construiți și afișați pe ecran, linie cu linie o matrice $n \times m$ construită după regula: $a[i][j] = \min(i, j)$. Numerotarea liniilor, respectiv coloanelor matricei începe de la 1.

Exemplu: Pentru $n=2$ și $m=3$ se va afișa matricea:

```
1 1 1
1 2 2
```

23. Fișierul text “graf.in” conține pe prima linie un număr natural n reprezentând numărul de vârfuri ale unui graf neorientat, iar 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.

a) Să se scrie o funcție **grad** ce primește ca parametru un număr natural x și returnează gradul vârfului x .

- b) Să se scrie programul C++ care citește datele din fișier și care afișează în fișirul text „graf.out”, pe primul rând, separate prin câte un spațiu vârfurile terminale ale grafului, sau mesajul „Nu există”, dacă în graf nu sunt vârfuri terminale, folosind apeluri utile ale subprogramului grad.

Exemplu: Dacă fișierul „graf.in” are forma:

```
5
0 0 1 0 1
0 0 0 1 1
1 0 0 0 0
0 1 0 0 0
1 1 0 0 0
```

atunci fișierul „graf.out” va conține numerele 3 și 4.

24. Fișierul text „muchii.txt” conține pe prima linie un număr natural n reprezentând numărul de vârfuri ale unui graf neorientat, pe al doilea rând un număr natural m reprezentând numărul de muchii ale unui graf neorientat, iar pe fiecare din următoarele m rânduri câte două numere naturale, separate printr-un spațiu, reprezentând extremitățile unei muchii a grafului.

a) Să se scrie o funcție **grad** ce primește ca parametru un număr natural x și returnează gradul vârfului x .

b) Să se scrie programul C++ care citește datele din fișier, construiește matricea de adiacență asociată grafului, și care afișează în fișirul text „grade.txt”, separate prin câte un spațiu vârfurile de grad maxim ale grafului, folosind apeluri utile ale subprogramului grad.

Exemplu: Dacă fișierul „muchii.txt” are forma:

```
5
6
1 2
2 3
2 4
3 4
3 5
4 5
```

atunci fișierul „grade.txt” va conține numerele 2, 3 și 4.

25. Fișierul text „adiacenta.in” conține pe prima linie un număr natural n reprezentând numărul de noduri ale unui graf orientat, iar 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.

a) Să se scrie o funcție **grad_intern** ce primește ca parametru un număr natural x și returnează gradul intern al nodului x .

Funcția **grad_extern** primește ca parametru un număr natural x și returnează gradul extern al nodului x .

b) Să se scrie programul C++ care citește datele din fișier, și care afișează în fișirul text „noduri.out”, separate prin câte un spațiu nodurile grafului care au gradul intern egal cu gradul extern, folosind apeluri utile ale subprogramelor **grad_intern** și **grad_extern**.

Exemplu: Dacă fișierul „adiacenta.in” are forma:

```

5
0 1 1 1 0
0 0 0 0 1
0 1 0 0 0
0 0 0 0 0
1 0 0 0 0

```

atunci fișierul „noduri.out” va conține numerele 3 și 5.

26. Fișierul text „arce.txt” conține pe prima linie un număr natural n reprezentând numărul de noduri ale unui graf orientat, pe al doilea rând un număr natural m reprezentând numărul de arce ale unui graf orientat, iar pe fiecare din următoarele m rânduri câte două numere naturale separate prin spații, reprezentând arcele corespunzătoare unui graf orientat.

a) Să se scrie o funcție *grad_intern* ce primește ca parametru un număr natural x și returnează gradul intern al nodului x .

Funcția *grad_extern* primește ca parametru un număr natural x și returnează gradul extern al nodului x .

b) Să se scrie programul C++ care citește datele din fișier, construiește matricea de adiacență asociată grafului, și care afișează în fișierul text „izolate.txt”, separate prin câte un spațiu nodurile izolate ale grafului, sau mesajul „Nu există”, dacă în graf nu sunt noduri izolate, folosind apeluri utile ale subprogramelor *grad_intern* și *grad_extern*.

Exemplu: Dacă fișierul „arce.txt” are forma:

```

5
3
1 2
2 1
2 4

```

atunci fișierul „izolate.txt” va conține numerele 3 și 5.

27. Se citesc numere naturale din fișierul text „numere.txt”. Să se afișeze numerele care au proprietatea de palindrom (numărul citit de la dreapta la stânga este egal cu numărul citit de la stânga la dreapta) și suma cifrelor să fie pară. Se vor folosi: un subprogram **palin** care va implementa un algoritm de determinare a proprietății de număr palindrom și un subprogram care să calculeze suma cifrelor unui număr natural.

Exemplu: Pentru numerele 12, 12321, 565, 45, 18, 121 se va afișa 565, 121.

