

FISIERE TEXT 2008

1. 3. Fisierul text **bac.txt** conține, pe o singură linie, cel mult **1000** de numere naturale nenule cu cel mult **4** cifre fiecare, numerele fiind separate prin câte un spațiu. Scrieți un program **C/C++** care citește de la tastatură un număr natural nenul **n** ($n \leq 999$) și numerele din fișierul **bac.txt** și care afișează pe ecran, separate prin câte un spațiu, toate numerele din fișier care sunt divizibile cu **n**. Dacă fișierul nu conține niciun astfel de număr, atunci se va afișa pe ecran mesajul **NU EXISTA**.
- Exemplu:** dacă fișierul **bac.txt** conține numerele: **3 100 40 70 25 5 80 6 3798**, pentru **n=10** atunci pe ecran se va afișa: **100 40 70 80** (10p.)
2. 3. Fișierului text **NR.TXT** conține pe o singură linie, separate prin câte un singur spațiu, cel mult **100** de numere **întregi**, fiecare număr având cel mult **4** cifre. Scrieți un program **C/C++** care citește numerele din fișierul **NR.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 fișierul **bac.txt** conține numerele: **-3 -10 0 7 -5 7 51 -800 6 3798**, atunci pe ecran se va afișa: **6 7 7 51 3798** (10p.)
3. Fișierului text **NR.TXT** conține pe o singură linie, separate prin câte un singur spațiu, cel mult **100** de numere naturale, fiecare număr având cel mult **4** cifre. Scrieți un program **C/C++** care citește toate numerele din fișierul **NR.TXT** și afișează pe ecran, separate prin câte un spațiu, în ordine crescătoare, toate numerele din fișier care au cel puțin **3** cifre. Dacă fișierul nu conține astfel de numere se va afișa pe ecran mesajul **NU EXISTA**. (10p.)
4. Fișierul text **NR.TXT** conține pe o singură linie, separate prin câte un singur spațiu, cel mult **100** de numere naturale, fiecare număr având cel mult **4** cifre. Scrieți un program **C/C++** care citește numerele din fișierul **NR.TXT** și afișează pe ecran, separate prin câte un spațiu, în ordine descrescătoare, toate numerele din fișier care au cel mult **2** cifre. Dacă fișierul nu conține astfel de numere se va afișa pe ecran mesajul **NU EXISTA**. (10p.)
5. Scrieți un program **C/C++** care citește de la tastură un număr natural **n** cu cel mult **8** cifre ($n \geq 10$) și care creează fișierul text **NR.TXT** ce conține numărul **n** și toate prefixele nenule ale acestuia, pe o singură linie, separate prin câte un spațiu, în ordine descrescătoare a valorii lor.
- Exemplu:** pentru **n=10305** fișierul **NR.TXT** va conține numerele:
10305 1030 103 10 1 (10p.)

6.

Se consideră fișierul **BAC.TXT** ce conține un sir **crescător** cu cel mult un milion de numere naturale de cel mult nouă cifre fiecare, separate prin câte un spațiu.

a) Să se scrie un program **C/C++** care, folosind un algoritm eficient din punct de vedere al memoriei utilizate și al timpului de executare, citește din fișier toti termenii șirului și afișează pe ecran, pe o singură linie, fiecare termen distinct al șirului urmat de numărul de aparitii ale acestuia în sir. Valorile afișate sunt separate prin câte un spațiu.

Exemplu: dacă fișierul **BAC.TXT** are următorul conținut:

1 1 1 5 5 5 5 9 9 11 20 20 20

programul va afișa:

1 3 5 4 9 2 11 1 20 3

deoarece 1 apare de 3 ori, 5 apare de 4 ori, etc.

(6p.)

b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența ei (3 – 4 rânduri).

(4p.)

7.

Scrieti un program **C/C++** care citește de la tastatură un număr natural **n** ($0 < n \leq 100$) și cele $3 * n$ elemente ale tabloului unidimensional **v**, fiecare element fiind un număr natural cu cel mult patru cifre fiecare. Tabloul este împărțit în trei zone cu câte **n** elemente: prima zonă conține primele **n** elemente din tablou, a doua zonă conține următoarele **n** elemente din tablou, restul elementelor fiind în zona a treia. Programul va interschimba primul element par (dacă există) al zonei **unu** cu ultimul element impar (dacă există) al zonei **trei** și apoi va scrie pe prima linie a fișierului text **BAC.TXT** toate elementele tabloului, separate prin câte un spațiu. În cazul în care unul dintre aceste două elemente, care urmează a fi interschimbat, nu există, programul nu va efectua nici o modificare asupra tabloului dat.

Exemplu: pentru **n=3** și **v= (1 2 3 4 5 6 7 8 9)**, fișierul **BAC.TXT** va conține:

1 9 3 4 5 6 7 8 2

(10p.)

8.

Se consideră două tablouri unidimensionale **a** și **b** fiecare având numere naturale de maximum patru cifre, **ordonate crescător**. Tabloul **a** conține **n** ($1 < n < 100$) numere pare, iar tabloul **b** conține **m** ($1 < m < 100$) numere impare.

a) Scrieți un program **C/C++** care citește de la tastatură valoarea lui **n** și cele **n** elemente ale tabloului **a**, apoi valoarea lui **m** și cele **m** elemente ale tabloului **b** după care scrie în fișierul **BAC.TXT** un număr maxim de elemente ale tablourilor date, numerele fiind scrise în ordine crescătoare, separate prin câte un spațiu, iar cele aflate pe poziții consecutive fiind de paritate diferită. Programul va utiliza un algoritm eficient din punct de vedere al timpului de executare.

Exemplu: pentru **n=6, m=5** și tablourile **a= (2, 4, 8, 10, 14, 16)** și **b= (3, 5, 7, 11, 15)** fișierul **BAC.TXT** va avea următorul conținut : 2 3 4 5 8 11 14 15 16

(6p.)

b) Descrieți succint, în limbaj natural, algoritmul pe baza căruia a fost scris programul de la punctul **a**), explicând în ce constă eficiența metodei utilizate.

(4p.)

9.

Se consideră fișierul **BAC.TXT** ce conține cel mult un milion de numere naturale separate prin spații, fiecare număr având cel mult nouă cifre.

a) Scrieți un program **C/C++** care citește toate numerele din fișierul **BAC.TXT** și determină, folosind un algoritm eficient din punct de vedere timpului de executare, cele mai mari două numere de trei cifre care nu se află în fișier. Dacă fișierul conține toate numerele de câte trei cifre atunci programul va afișa pe ecran valoarea 0.

Exemplu: dacă fișierul **BAC.TXT** conține numerele:

12 2345 123 67 989 6 999 123 67 989 999

atunci programul va afișa

998 997

(6p.)

b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența ei (3 – 4 rânduri).

(4p.)

10.

Evidența produselor vândute de o societate comercială este păstrată în fișierul **PRODUSE.TXT**. Pentru fiecare produs se cunoaște tipul produsului (un număr natural de cel mult 4 cifre), cantitatea exprimată în kilograme (un număr natural mai mic sau egal cu 100) și prețul unui kilogram (un număr natural mai mic sau egal cu 100).

Produsele de același tip pot fi vândute în cantități diferite, fiecare vânzare fiind înregistrată separat.

Fișierul **PRODUSE.TXT** are cel mult 200000 de linii și fiecare linie conține trei numere naturale, separate prin câte un spațiu, ce reprezintă, în această ordine tipul, cantitatea și prețul de vânzare al unui produs la un moment dat.

a) Să se scrie un program c/c++, care utilizând un algoritm eficient din punct de vedere al timpului de executare, determină pentru fiecare tip de produs vândut suma totală obținută în urma vânzărilor. Programul va afișa pe câte o linie a ecranului tipul produsului și suma totală obținută, separate prin câte un spațiu, ca în exemplu.

Exemplu: dacă fișierul **PRODUSE.TXT** are conținutul alăturat, programul va afișa numerele următoare:

1 150
2 30
3 5

3	1	5
1	20	5
2	10	3
1	10	5

(6p.)

b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența (3 - 4 rânduri). (4p.)

11.

Fișierul text **numere.txt** conține pe prima sa linie un număr natural **n** ($n < 30000$), iar pe a doua sa linie, **n** numere întregi, având maximum 4 cifre fiecare. Se cere să se afișeze pe ecran un sir de **n** numere întregi, cu proprietatea că valoarea termenului de pe poziția **i** ($i=1, 2, \dots, n$) din acest sir este egală cu cea mai mare dintre primele **i** valori de pe a doua linie a fișierului **numere.txt**.

a) Descrieți pe scurt un algoritm de rezolvare, eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat, explicând în ce constă eficiența sa. (4p.)

b) Scrieți programul c/c++ corespunzător algoritmului descris. (6p.)

Exemplu: dacă fișierul **numere.txt** are conținutul alăturat, se afișează pe ecran numerele

4 6 6 7 8 8 8 8 9 10 10

12											
4	6	3	7	8	1	6	2	7	9	10	8

12.

Fișierele text **NR1.TXT** și **NR2.TXT** conțin, separate prin câte un spațiu, mai multe numere întregi de cel mult 9 cifre fiecare. Fiecare dintre fișiere conține cel mult 100 de valori și numerele din fiecare fișier sunt ordonate strict crescător. Se cere să se afișeze pe ecran, în ordine crescătoare, numerele divizibile cu 5 care se găsesc doar în unul din cele două fișiere.

Exemplu: dacă fișierul **NR1.TXT** conține numerele 1 2 3 4 7 20 60, iar fișierul **NR2.TXT** conține numerele 3 5 7 8 9 10 12 20 24, atunci se vor afișa pe ecran valorile 5 10 60.

a) Descrieți un algoritm de rezolvare a acestei probleme, eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat, explicând în ce constă eficiența acestuia. (4p.)

b) Scrieți programul c/c++ corespunzător algoritmului descris. (6p.)

13.

Pe prima linie a fișierului text **BAC.IN** se găsesc, separate prin câte un spațiu, mai multe numere naturale de cel mult 9 cifre fiecare. Scrieți un program c/c++ care citește numerele din acest fișier, elimină toate cifrele impare din fiecare dintre aceste numere și apoi scrie în fișierul text **BAC.OUT** numerele astfel obținute. Dacă un număr din fișierul **BAC.IN** conține doar cifre impare și cifra 0, acesta nu va mai apărea deloc în fișierul de ieșire. (10p.)

Exemplu: dacă fișierul **BAC.IN** conține numerele 25 7 38 1030 45127 0 35 60 15 atunci **BAC.OUT** va avea conținutul: 2 8 42 60.

14. Fișierul text **BAC.TXT** conține mai multe numere naturale, cu cel mult 6 cifre fiecare, câte un număr pe fiecare linie a fișierului.
Scrieți un program C/C++ care citește toate numerele din fișierul **BAC.TXT** și le afișează pe ecran, câte 5 pe fiecare linie, separate prin câte un spațiu, cu excepția ultimei linii care poate conține mai puțin de 5 numere. Programul va afișa apoi pe ecran, pe o linie separată, câte numere din fișier au suma cifrelor pară.

11
21
30
40
51
16
17
10
1

Exemplu: dacă fișierul are conținutul alăturat, pe ecran se vor afișa numerele de mai jos:
 11 21 30 40 51
 16 17 10 1
 4

(10p.)

15. În fișierul text **BAC.IN** se găsesc, pe o singură linie, separate prin câte un spațiu, mai multe numere naturale de cel mult 6 cifre fiecare. Se cere să se determine și să se afișeze pe ecran, separate printr-un spațiu, ultimele **două** numere prime (nu neapărat distinse) din fișierul **BAC.IN**. Dacă în fișier se găsește un singur număr prim sau niciun număr prim se va scrie pe ecran mesajul **Numere prime insuficiente**.

Exemplu: dacă fișierul **BAC.IN** conține valorile: 12 5 68 13 8 17 9 31 42 se va afișa 17 31.

a) Descrieți în limbaj natural un algoritm eficient, din punct de vedere al spațiului de memorie și al timpului de executare, pentru rezolvarea acestei probleme, explicând în ce constă eficiența acestuia. (4p.)

b) Scrieți programul C/C++ corespunzător algoritmului descris. (6p.)

16. În fișierul **numere.txt** sunt memorate maximum 10000 de numere naturale cu cel mult 9 cifre fiecare. Fiecare linie a fișierului conține câte un număr. Se cere afișarea pe ecran, în ordine descrescătoare, a tuturor cifrelor care apar în numerele din fișier. Alegeti un algoritm de rezolvare eficient din punct de vedere al memoriei utilizate și al timpului de execuție.

Exemplu: dacă fișierul **numere.txt** conține:
 267
 39628
 79
 se va tipări 9987766322.

a) Descrieți succint, în limbaj natural, strategia de rezolvare și justificați eficiența algoritmului ales. (4p.)

b) Scrieți programul C/C++ corespunzător algoritmului ales. (6p.)

17. În fișierul **numere.txt** pe prima linie este memorat un număr natural **n** ($n \leq 10000$), iar pe linia următoare un sir de **n** numere naturale distincte două câte două, separate prin câte un spațiu, cu maximum 4 cifre fiecare. Se cere afișarea pe ecran a poziției pe care s-ar găsi primul element din sirul aflat pe linia a doua a fișierului, în cazul în care sirul ar fi ordonat crescător. Numerotarea pozițiilor elementelor în cadrul sirului este de la 1 la **n**. Alegeti un algoritm de rezolvare eficient din punct de vedere al memoriei utilizate și al timpului de execuție.

Exemplu: dacă fișierul **numere.txt** conține:
 6
 267 13 45 628 7 79
 se va afișa 5, deoarece primul element din sirul inițial, 267, s-ar găsi pe poziția a cincea în sirul ordonat crescător.

a) Descrieți succint, în limbaj natural, strategia de rezolvare și justificați eficiența algoritmului ales. (4p.)

b) Scrieți programul C/C++ corespunzător algoritmului ales. (6p.)

18.

În fișierul **numere.txt** este memorat un sir de maximum **10000** numere naturale, distincte două câte două, cu maximum **4** cifre fiecare, separate prin câte un spațiu. Pentru un număr **k** citit de la tastatură, se cere afișarea pe ecran a poziției pe care se va găsi acesta în sirul de numere din fișier, dacă sirul ar fi ordonat descrescător, sau mesajul **nu există**, dacă numărul **k** nu se află printre numerele din fișier. Alegeți un algoritm eficient de rezolvare din punct de vedere al memoriei utilizate și al timpului de execuție.

Exemplu: dacă fișierul **numere.txt** conține numerele **26 2 5 30 13 45 62 7 79**, iar **k** are valoarea **13**, se va afișa **6** deoarece **13** s-ar găsi pe poziția a **șasea** în sirul ordonat descrescător.

a) Descrieți succinct, în limbaj natural, strategia de rezolvare și justificați eficiența algoritmului ales. (4p.)

b) Scrieți programul **c/c++** corespunzător algoritmului ales. (6p.)

19.

În fiecare dintre fișierele **nr1.txt** și **nr2.txt** este memorată pe prima linie câte o valoare naturală **n** de cel mult **8** cifre, iar pe linia următoare sunt memorate câte **n** numere naturale, cu maximum **4** cifre fiecare, ordonate strict crescător și separate prin câte un spațiu. Se cere afișarea pe ecran, separate prin câte un spațiu, în ordine strict crescătoare, a tuturor numerelor aflate pe a a doua linie în cel puțin unul dintre cele două fișiere. În cazul în care un număr apare în ambele fișiere, el va fi afișat o singură dată. Alegeți un algoritm de rezolvare eficient din punct de vedere al memoriei utilizate și al timpului de execuție.

Exemplu: pentru următoarele fișiere:

nr1.txt

5

3 6 8 9 12

se va afișa **2 3 5 6 7 8 9 12 13**.

nr2.txt

6

2 3 5 7 9 13

a) Descrieți succinct, în limbaj natural, strategia de rezolvare și justificați eficiența algoritmului ales. (4p.)

b) Scrieți programul **c/c++** corespunzător algoritmului ales. (6p.)

20.

În fiecare dintre fișierele **nr1.txt** și **nr2.txt** este memorată pe prima linie câte o valoare naturală **n** de cel mult **8** cifre, iar pe linia următoare sunt memorate câte **n** numere naturale, cu maximum **4** cifre fiecare, ordonate strict crescător și separate prin câte un spațiu. Se cere afișarea pe ecran, separate prin câte un spațiu, în ordine strict crescătoare, a tuturor numerelor aflate pe a a doua linie atât în primul cât și în al doilea fișier. Alegeți un algoritm de rezolvare eficient din punct de vedere al memoriei utilizate și al timpului de execuție.

Exemplu: pentru următoarele fișiere:

nr1.txt

5

3 6 8 9 12

nr2.txt

6

2 3 5 7 9 13

se va afișa **3 9**.

a) Descrieți succinct, în limbaj natural, strategia de rezolvare și justificați eficiența algoritmului ales. (4p.)

b) Scrieți programul **c/c++** corespunzător algoritmului ales. (6p.)

21.

Fișierul **BAC.TXT** conține pe prima linie două numere naturale **n** și **k** separate de un spațiu ($3 \leq n \leq 10000$, $2 \leq k \leq n/2$), iar pe a doua linie un sir de **n** numere naturale x_1, x_2, \dots, x_n separate prin câte un spațiu, fiecare număr din acest sir având cel mult patru cifre.

a) Scrieți un program **C/C++** care citește numerele din fișier și determină, utilizând o metodă eficientă din punct de vedere al timpului de executare, cel mai mic indice **i** ($1 \leq i \leq n-k+1$) pentru care media aritmetică a numerelor $x_1, x_{i+1}, \dots, x_{i+k-1}$ este maximă. Programul afișează valoarea lui **i** pe ecran.

Exemplu: pentru fișierul alăturat se afișează **2**, deoarece media maximă se obține pentru **9, 4, 7**. (6p.)

b) Explicați succint, în limbaj natural, metoda utilizată la punctul a, justificând eficiența acesteia. (4p.)

22.

scrieți programul c/c++ care citește din fișierul **BAC.TXT** numărul întreg **n** ($1 \leq n \leq 10000$) și un sir de **n** perechi de numere întregi **a b** ($1 \leq a \leq b \leq 32000$), fiecare pereche fiind scrisă pe o linie nouă a fișierului, cu un spațiu între cele două numere. Programul afișează pe ecran pentru fiecare pereche **a, b** cel mai mare număr natural din intervalul închis **[a, b]** care este o putere a lui 2 sau numărul 0 dacă nu există nicio putere a lui 2 în intervalul respectiv.

Exemplu: dacă fișierul **BAC.TXT** conține numerele

3
2 69
10 20
19 25

se va afișa: 64 16 0.

(10p.)

23.

Fișierul **BAC.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 C/C++ care determină intervalele care au proprietatea că intersecția cu oricare dintre celelalte **n-1** intervale este vidă și afișează pe câte o linie a ecranului, separate printr-un spațiu, numerele care reprezintă capetele intervalelor determinate.

(10p.)

Exemplu: dacă fișierul **BAC.TXT** are conținutul alăturat, pe ecran se va

afișa:
2 6
17 20

4
17 20
2 6
10 15
8 16

24.

Fișierul **bac.txt** conține pe prima linie numărul natural **n**, $1 \leq n \leq 30000$, pe următoarele **n** linii un tablou unidimensional de **n** numere întregi, ordonate crescător, iar pe ultima linie două numere întregi **a** și **b** ($a \leq b$) separate de un spațiu. Fiecare dintre cele **n** numere, precum și valorile **a** și **b**, au cel mult patru cifre.

a) Scrieți un program c/c++, eficient din punct de vedere al timpului de executare, care afișează pe ecran cel mai mic număr întreg din intervalul închis **[a, b]** care se găsește în tabloul dat. Dacă nu există un astfel de număr programul afișează textul **NU**.

Exemplu: dacă fișierul **bac.txt** are conținutul alăturat, programul afișează | 4
valoarea 11 | -2

b) Descrieți în limbaj natural metoda utilizată și explicați în ce constă eficiența ei. | 7
| 11
| 35
| 8 15

25.

Fișierul text **NUMAR.TXT** conține pe prima linie un număr real pozitiv **x** care are cel mult două cifre la partea întreagă și cel mult **șapte** cifre după punctul zecimal..

a) Scrieți un program c/c++ care, utilizând un algoritm eficient din punct de vedere al timpului de executare și al memoriei utilizate, afișează pe ecran separate printr-un spațiu, două numere naturale al căror raport este egal cu **x** și a căror diferență absolută este minimă.

Exemplu: dacă fișierul conține valoarea alăturată, se vor afișa pe ecran | 0.375
numerele 3 8. | 6p.)

b) Descrieți în limbaj natural metoda utilizată și explicați în ce constă eficiența ei. | 4p.)

26.

a) Scrieți definiția completă a subprogramului **sterge**, care primește prin cei 4 parametri **v, n, i, j**:

- **v**, un tablou unidimensional cu maximum 100 de elemente întregi din intervalul [-1000;1000]

- **n**, un număr natural reprezentând numărul de elemente din tabloul **v**

- **i** și **j** două valori naturale cu $1 \leq i \leq j \leq n$

și elimină din tabloul **v** elementele v_1, v_{i+1}, \dots, v_j actualizând valoarea parametrului **n**. (6p.)

b) Fișierul **NUMERE.IN** conține pe prima linie un număr natural nenul **n** ($1 \leq n \leq 100$) și pe următoarea linie **n** numere întregi din intervalul [-1000;1000], separate prin câte un spațiu. Scrieți un program c/c++ care citește din fișierul **NUMERE.IN** numărul natural **n**, construiește în memorie un tablou unidimensional **v** cu cele **n** numere întregi aflate pe linia a doua în fișier și utilizează apeluri utile ale subprogramului **sterge** pentru a elimina din tablou un număr minim de elemente astfel încât să nu existe două elemente alăturate cu aceeași valoare. Elementele tabloului obținut se afișează pe ecran, separate prin câte un spațiu.

Exemplu: Dacă fișierul **NUMERE.IN** are conținutul:

12

10 10 2 2 19 9 9 9 15 15 15 atunci se afișează 10 2 19 9 15. (10p.)

27.

Fișierul **NUMERE.IN** conține pe prima linie un număr natural nenul **n** ($2 \leq n \leq 100$) și pe următoarea linie **n** numere reale pozitive în ordine strict crescătoare separate prin câte un spațiu.

a) Scrieți un program c/c++ care, utilizând un algoritm eficient din punct de vedere al memoriei utilizate, determină și afișează pe ecran cel mai mare număr natural **x** cu proprietatea că în orice interval deschis având capete oricare două dintre cele **n** numere aflate pe linia a doua în fișierul **NUMERE.IN** se găsesc cel puțin **x** numere întregi. Numărul astfel determinat se afișează pe ecran.

Exemplu: dacă fișierul **NUMERE.IN** are conținutul:

6

3.5 5.1 9.2 16 20.33 100 atunci se afișează 2 pentru că în oricare dintre intervalele $(3.5; 5.1)$, $(3.5; 9.2)$, $(3.5; 16)$, $(3.5; 20.33)$, $(3.5; 100)$, $(5.1; 9.2)$, $(5.1; 16)$, $(5.1; 20.33)$, $(5.1; 100)$, $(9.2; 16)$, $(9.2; 20.33)$, $(9.2; 100)$, $(16; 20.33)$, $(16; 100)$, $(20.33; 100)$ există cel puțin două numere întregi. (6p.)

b) Descrieți în limbaj natural metoda utilizată și explicați în ce constă eficiența ei. (4p.)

28.

a) Scrieți definiția completă a unui subprogram **primul**, care

- primește prin singurul său parametru, **a**, o valoare naturală din intervalul [2,10000]

- returnează o valoare naturală reprezentând cel mai mic divizor al numărului **a** mai mare strict decât 1. (6p.)

b) Fișierul **NUMERE.IN** conține pe prima linie un număr natural nenul **n** ($1 \leq n \leq 100$) și pe următoarea linie **n** numere naturale din intervalul [2,10000] separate prin câte un spațiu.

Un număr natural **n** se numește „**aproape prim**” dacă este egal cu produsul a două numere prime distincte. De exemplu, numărul 14 este „**aproape prim**” pentru că este egal cu produsul numerelor prime 2 și 7.

Scrieți un program c/c++ care determină, folosind apeluri utile ale suprogramului **primul**, cel mai mare număr „**aproape prim**” de pe linia a doua a fișierului **NUMERE.IN**. În caz afirmativ se afișează pe ecran mesajul DA urmat de numărul determinat iar în caz contrar mesajul NU.

Exemplu: dacă fișierul **NUMERE.IN** are conținutul:

6

100 14 21 8 77 35

atunci se afișează pe ecran DA 77 pentru că numărul 77 este cel cel mai mare dintre numerele „**aproape prime**” din fișier ($14=2 \cdot 7$, $21=3 \cdot 7$, $77=7 \cdot 11$, $35=5 \cdot 7$). (10p.)

29.

Se consideră două tablouri unidimensionale **A** și **B** cu elemente numere naturale din intervalul $[1;10000]$. Spunem că tabloul **A** **"se poate reduce"** la tabloul **B** dacă există o împărțire pe secvențe de elemente aflate pe poziții consecutive în tabloul **A** astfel încât prin înlocuirea secvențelor cu suma elementelor acestora să se obțină, în ordine, elementele tabloului **B**.

De exemplu tabloul

7	3	4	1	6	4	6	9	7	1	8	7
14			26			16					

se poate reduce la tabloul

Fisierul **NUMERE.IN** conține pe prima linie două numere naturale nenule **n** și **m** ($1 \leq n \leq 100$), pe linia a doua **n** numere naturale din intervalul $[1;10000]$ și pe linia a treia alte **m** numere naturale din intervalul $[1;10000]$. Pe fiecare linie numerele sunt separate prin câte un spațiu.

a) Scrieți un program c/c++ care citește cele două numere naturale **n** și **m** din fisierul **NUMERE.IN**, construiește în memorie două tablouri unidimensionale **A** și **B** cu elementele aflate în fisier pe a doua, respectiv a treia linie și verifică, utilizând un algoritm eficient din punct de vedere al timpului de executare, dacă tabloul **A** se poate reduce la tabloul **B**. Programul afișează pe ecran mesajul **DA** în caz afirmativ și mesajul **NU** în caz negativ. (6p.)

b) Descrieți în limbaj natural metoda utilizată și explicați în ce constă eficiența ei. (4p.)

30.

Fisierul **NUMERE.IN** conține pe prima linie un număr natural nenul **n** ($1 \leq n \leq 100$) și pe următoarea linie **n** numere reale pozitive **ordonate crescător**, separate prin câte un spațiu.

a) Scrieți un program c/c++ care citește din fisierul **NUMERE.IN** numărul natural **n**, și determină, utilizând un algoritm eficient din punct de vedere al timpului de executare și al memoriei utilizate, numărul **minim** de intervale închise de forma $[x; x+1]$, cu **x** număr natural, a căror reuniune include toate numerele reale din fisier.

Exemplu: Dacă fisierul **NUMERE.IN** are conținutul:

6

2.3 2.8 5.1 5.7 5.9 6.3 atunci se afișează 3 (intervalele $[2; 3]$, $[5; 6]$, $[6; 7]$ sunt cele 3 intervale de forma cerută care conțin numere din sir). (6p.)

b) Descrieți în limbaj natural metoda utilizată și explicați în ce constă eficiența ei. (4p.)

31.

În fisierul **numere.txt**, se află memorate, pe prima linie un număr natural **n** ($1 \leq n \leq 100$), iar pe fiecare dintre următoarele **n** linii câte două numere întregi **x, y** ($-100 \leq x \leq y \leq 100$) reprezentând capetele câte unui segment $[x, y]$ desenat pe axa **ox** de coordonate.

a) Scrieți în limbajul c/c++ un program eficient din punct de vedere al timpului de executare și al spațiului de memorare, care citește din fisier datele existente, determină segmentul rezultat în urma intersecției tuturor celor **n** segmente date și afișează pe ecran două numere despărțite printr-un spațiu ce reprezintă capetele segmentului cerut. Dacă segmentele nu au nici un punct comun se va afișa pe ecran valoarea 0. (6p.)

b) Descrieți în limbaj natural algoritmul utilizat, justificând eficiența acestuia. (4p.)

Exemplu: dacă fisierul **numere.txt** are conținutul alăturat, se va afișa pe ecran

5
-7 10
3 20
-5 5
0 12
-8 30

32.

În fisierul **numere.txt** sunt memorate pe mai multe linii, numere întregi (cel mult 100), numerele de pe aceeași linie fiind despărțite prin câte un spațiu, fiecare număr având cel mult 9 cifre. Să se determine cele mai mici două valori având **exact** două cifre fiecare, memorate în fisier și să se afișeze pe ecran aceste valori, despărțite prin câte un spațiu.

a) Descrieți în limbaj natural o metodă eficientă de rezolvare din punct de vedere al gestionării memoriei și timpului de executare. (4p.)

b) Scrieți programul c/c++ corespunzător metodei descrise la punctul a. (6p.)

Exemplu: dacă fisierul **numere.txt** are conținutul alăturat, se va afișa pe ecran

5 10
3 -77 20
50 5 0 12 18 30

33.

34.

c) În fișierul **numere.txt**, se află memorat pe prima linie un număr natural n ($n < 100$), iar pe fiecare dintre următoarele n linii, câte n numere întregi cu cel mult 4 cifre fiecare. Scrieți programul c/c++ care citește din fișier datele existente, determină liniile din fișier pe care sunt memorat în ordine termenii unei progresii aritmetice și afișează pe ecran, folosind apeluri ale subprogramului **max** cel mai mare număr (diferit de cel situat pe prima linie) din fișier, care în plus este termenul unei progresii aritmetice. **(10p.)**

Exemplu: dacă fișierul **numere.txt** are conținutul alăturat, se va afișa 50, deoarece progresiile aritmetice sunt:

(-9 -7 -5 -3 -1),
(50 40 30 20 10) și
(18 17 16 15 14)

5
5 7 3 1 9
-9 -7 -5 -3 -1
2 5 8 14 11
50 40 30 20 10
18 17 16 15 14

35.

b) Scrieți programul c/c++ care citește de la tastatură un număr natural n ($0 < n < 100$), apoi n numere naturale (cu cel mult 4 cifre fiecare). Programul determină, folosind apeluri utile ale subprogramului **sum**, pentru fiecare număr natural, suma divizorilor săi proprii și afișează pe ecran sumele determinate, în ordinea crescătoare a valorilor lor, separate prin câte un spațiu. **(6p.)**

Exemplu: dacă $n=5$ și numerele citite sunt 10 2 33 6 11

valorile afișate pe ecran vor fi: 0 0 5 7 14

deoarece suma divizorilor lui 10 este 7, suma divizorilor lui 2 este 0, suma divizorilor lui 33 este 14, suma divizorilor lui 6 este 5, suma divizorilor lui 11 este 0.

36.

b) Pe prima linie a fișierului **bac.in** se află un număr natural nenul $n \leq 15000$, iar pe a doua linie a fișierului se află un sir de n numere naturale, despărțite prin câte un spațiu, fiecare număr fiind format din cel mult 4 cifre.

Scrieți un program c/c++ care citește numerele din fișier și afișează pe ecran, folosind apeluri utile la subprogramului **cifra**, cel mai mare număr care se poate forma cu ultimele cifre pare ale fiecărui element, dacă acestea există. Alegeti o metodă de rezolvare eficientă ca timp de executare.

Exemplu: dacă fișierul **bac.in** are conținutul alăturat pe ecran se va afișa: 64220 (6p.) 369 113 2 0 33 1354 42

a) Scrieți doar antetul subprogramului **d1v** care primește prin intermediul parametrului **x** un număr natural nenul cu cel mult 4 cifre, și returnează numărul de divizori primi ai lui **x**. **(4p.)**

b) Pe prima linie a fișierului **bac.in** se află un număr natural nenul $n \leq 1000$, iar pe a doua linie a fișierului se află un sir format din n numere naturale nenule, despărțite prin câte un spațiu, fiecare număr fiind format din cel mult 4 cifre. Scrieți un program c/c++ care citește numerele din fișier și care afișează pe ecran, folosind apeluri utile ale subprogramului **d1v**, prima și ultima valoare din sirul celor n numere citite, care au un număr par de divizori primi.

Exemplu: dacă fișierul **bac.in** are conținutul alăturat pe ecran se va afișa: 20 10 (6p.) 30 105 20 140 7 10 5

38.

Se consideră subprogramul **inter** cu doi parametri: **x** și **y** (numere întregi formate din cel mult patru cifre fiecare); subprogramul interschimbă valorile a două variabile transmise prin intermediul parametrilor **x** și **y**.

a) Scrieți în limbajul c/c++ definiția completă a subprogramului **inter**. **(4p.)**

b) Pe prima linie a fișierului **bac.in** se află un număr natural nenul $n \leq 1000$, iar pe a doua linie a fișierului se află un sir de n numere naturale nenule, despărțite prin câte un spațiu, fiecare număr fiind format din cel mult 4 cifre. Scrieți un program c/c++ care afișează pe ecran, în ordine crescătoare, numerele aflate pe a doua linie a fișierului. Numerele vor fi afișate pe o singură linie iar între două numere se va lăsa un spațiu. Se vor folosi apeluri utile ale subprogramului **inter**. **(6p.)**

39.

Se consideră subprogramul **pr** care primește prin intermediul parametrului **a** un număr natural nenul cu cel mult 9 cifre și returnează **1** dacă numărul este prim și **0** în caz contrar.

a) Scrieți numai antetul subprogramului **pr**. (4p.)

b) Considerăm un număr natural nenul $n > 99$ cu cel mult 9 cifre. Să se realizeze un program C/C++ care citește numărul **n** și care, folosind apeluri utile ale subprogramul **pr**, afișează pe ecran, separate prin câte un spațiu, doar valorile prime din sirul numerelor obținute din **n**, prin eliminarea succesivă a ultimei cifre, apoi a ultimelor două cifre, apoi a ultimelor trei cifre etc., până se obține un număr de două cifre, ca în exemplu.

Exemplu: pentru **n=193124** se obține sirul de valori **19312, 1931, 193, 19**. din care se vor afișa pe ecran doar valorile **1931 193 19**. (6p.)

40.

Pe prima linie a fișierului **bac.in** se află un număr natural nenul $n \leq 1000$, iar pe a doua linie a fișierului se află un sir format din **n** numere naturale, despartite prin câte un spațiu, fiecare număr fiind format din cel mult 4 cifre. Scrieți un program C/C++ care citește numerele din fișier și care afișează pe ecran mesajul **DA** dacă elementele pare în sir sunt în ordine crescătoare, iar cele impare sunt în ordine descrescătoare și mesajul **NU** în caz contrar.

Exemplu: dacă fișierul **bac.in** are conținutul **8**
alăturat pe ecran se va afișa: **DA (10p.) 10 1133 12 331 42 1354 221 13**

41.

Fișierul text **numere.txt** conține pe prima linie un număr natural **n** ($0 < n < 100000$) iar pe doua linie **n** numere naturale, formate dintr-o singură cifră, separate prin câte un spațiu.

a) Scrieți un program C/C++ care determină în mod eficient, din punct de vedere al timpului de executare, cea mai mare cifră dintre cele situate pe a doua linie a fișierului, precum și numărul de apariții ale acesteia. Cele două numere vor fi afișate pe o singură linie a ecranului, separate printr-un spațiu.

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

**7
3 5 2 1 5 3 1**

atunci pe ecran se va afișa: **5 2.** (6p.)

b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența ei (3 – 4 rânduri). (4p.)

42.

Fișierul text **numere.txt** conține pe prima linie un număr natural **n** ($0 < n < 100000$) iar pe doua linie, separate prin câte un spațiu, **n** numere naturale formate din cel mult 2 cifre fiecare.

a) Scrieți un program C/C++ care determină în mod eficient, din punct de vedere al timpului de executare, numerele ce apar o singură dată în a doua linie a fișierului. Aceste numere vor fi afișate pe ecran în ordine crescătoare, separate prin câte un spațiu.

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

**7
3 5 2 1 5 23 1**

atunci pe ecran se va afișa: **2 3 23.** (6p.)

b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența ei (3 – 4 rânduri). (4p.)

43.

Fișierul text **numere.txt** conține pe prima linie un număr natural **n** ($0 < n < 100000$) iar pe a doua linie **n** cifre, separate prin câte un spațiu.

a) Scrieți un program C/C++ care determină în mod eficient, din punct de vedere al timpului de executare, cel mai mare număr ce se poate forma cu toate cifrele conținute de a doua linie a fișierului **numere.txt**. Numărul determinat se va afișa pe ecran.

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

**7
2 5 3 1 5 8 9**

atunci pe ecran se va afișa: **9855321.** (6p.)

b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența ei (3 – 4 rânduri). (4p.)

44.

Fișierul text **numere.txt** conține pe prima linie un număr natural n ($0 < n < 100000$) iar pe a doua linie n numere naturale, formate din cel mult 4 cifre, separate prin câte un spațiu.

a) Scrieți un program C/C++ care determină în mod eficient, din punct de vedere al timpului de executare, cifrele ce apar în scrierea numerelor situate pe a doua linie a fișierului. Programul va afișa pe ecran aceste cifre în ordine crescătoare, separate prin câte un spațiu.

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

```
7
243 32 545 74 12 1344 90
```

atunci pe ecran se va afișa: 0 1 2 3 4 5 7 9 (6p.)

b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența ei (3 – 4 rânduri). (4p.)

45.

Fișierul text **numere.txt** conține pe prima linie un număr natural n ($0 < n < 100000$) iar pe a doua linie n numere naturale, formate din cel mult 2 cifre, separate prin câte un spațiu.

a) Scrieți un program C/C++, eficient atât din punct de vedere al timpului de executare, care afișează pe ecran toate numerele situate pe a doua linie a fișierului, în ordinea crescătoare a valorilor lor, separate prin câte un spațiu.

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

```
7
12 21 22 11 9 12 3
```

atunci pe ecran se va afișa: 3 9 11 12 12 21 22 (6p.)

b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența ei (3 – 4 rânduri). (4p.)

46.

Fișierul **bac.txt** conține pe prima linie un număr natural n ($n \leq 100$), iar pe a doua linie, separate prin câte un spațiu, n numere naturale nenule, cu cel mult 4 cifre fiecare. Scrieți programul C/C++ care citește de la tastatură un număr natural k ($k \leq 25$), construiește și afișează pe ecran un tablou unidimensional ce conține numerele citite din fișierul **bac.txt** care au cel puțin k divizori.

Exemplu: dacă $k = 9$ iar fișierul are conținutul
alăturat, atunci tabloul care se afișează este:
100 400 180 616 (10p.)

47.

Fișierul text **bac.txt** conține pe prima linie un număr natural n ($n < 100$), iar pe a doua linie, separate prin câte un spațiu, n numere naturale, mai mici decât 30000 fiecare.

Scrieți un program C/C++ care citește de la tastatură un număr natural k ($k < 10$) precum și numerele din fișierul **bac.txt** și determină și afișează pe ecran, cu câte un spațiu între ele, toate numerele de pe a doua linie a fișierului care conțin în scrierea lor cifra memorată în variabila k . Dacă nu există un asemenea număr pe a doua linie a fișierului, se afișează pe ecran mesajul **NU**.

Exemplu: dacă se citește de la tastatură $k=2$, iar fișierul are conținutul alăturat, atunci pe ecran se afișează numerele:
234 5678 317 809 202 427 92 6004
234 202 427 92 (10p.)

48.

Fișierul text **bac.in** conține pe prima linie un număr natural n ($0 < n < 5000$), iar pe a doua linie, separate prin câte un spațiu, n numere naturale, formate din cel mult 4 cifre fiecare.

Scrieți un program C/C++ care determină și scrie în fișierul **bac.out** toate numerele conținute de a doua linie a fișierului care apar o singură dată în această linie. Numerele determinate se vor afișa în ordinea crescătoare a valorilor lor, separate prin câte un spațiu.

Exemplu: dacă pe prima linie a fișierului **bac.in** se află 10, iar pe linia a două se găsesc numerele **2 4548 568 4548 57 89 5974 2 89 32** atunci valorile căutate sunt **32 57 568 5974**. (10p.)

49.

Se consideră subprogramul **cmax** care prin parametrul **a** primește un număr natural nenul mai mic decât 30000, iar prin parametrul **b** furnizează cifra maximă din numărul **a**.

a) Scrieți, folosind limbajul C/C++, doar antetul subprogramului **cmax**. (4p.)

b) Fișierul **bac.txt** conține cel mult 1000 numere naturale nenule, mai mici decât 30000 fiecare, separate prin câte un spațiu. Scrieți programul C/C++ care citește din fișierul **bac.txt** toate numerele și care determină cea mai mare cifră din scrierea lor folosind apeluri utile ale subprogramului **cmax**. Cifra determinată se va afișa pe ecran.

Exemplu: dacă fișierul **bac.txt** conține valorile: 23 12 64 12 72 345 67 23 71 634 atunci pe ecran se afișează 7. (10p.)

50.

Fișierul **bac.in** conține pe prima linie un număr natural **n** ($0 < n < 5000$), iar pe a doua linie, separate prin câte un spațiu, **n** numere naturale, formate din cel mult 4 cifre fiecare.

Scrieți un program C/C++ care determină și scrie în fișierul **bac.out**, toate numerele, citite de pe a doua linie a fișierului **bac.in**, care apar de cel puțin două ori. Numerele determinate se vor scrie în ordine crescătoare, pe aceeași linie, separate prin câte un spațiu.

Exemplu: dacă fișierul **bac.in** conține pe prima linie numărul 11, iar pe linia a doua valorile 23 12 54 12 78 345 67 23 78 934 23 atunci fișierul **bac.out** va conține: 12 23 78 (10p.)

51.

Fișierul text **date.in** conține pe prima linie un număr natural nenul **n** ($n \leq 100$), iar pe a doua linie **n** numere naturale nenule, separate prin câte un spațiu, fiecare număr având maximum 4 cifre. Scrieți un program C/C++ care citește toate numerele din fișierul text **date.in**, construiește în memorie un tablou unidimensional **a**, cu cele **n** elemente din fișier și afișează pe ecran perechile $(a[i], a[j])$, $1 \leq i < j \leq n$, cu proprietatea că elementele fiecărei dintre aceste perechi au aceeași paritate. Fiecare pereche se va afișa pe câte o linie a ecranului, elementele perechii fiind separate prin câte un spațiu. În cazul în care nu există nicio pereche se va afișa valoarea 0...

Exemplu: dacă fișierul **date.in** are conținutul alăturat, se vor afișa:

16 22

16 6

22 6

3 1

5
16 22 3 6 1

(10p.)

52.

Se consideră subprogramul **cmmdc**, care primește prin intermediul a doi parametri, **a** și **b**, două numere naturale nenule, cu maximum 8 cifre fiecare, și returnează cel mai mare divizor comun al valorilor parametrilor **a** și **b**.

a) Scrieți numai antetul subprogramului **cmmdc**. (4p.)

b) Fișierul text **date.in** conține pe prima linie un număr natural nenul **n** ($n \leq 100$), iar pe a doua linie un sir format din **n** numere naturale nenule, separate prin câte un spațiu, fiecare număr având maximum 8 cifre. Scrieți un program C/C++ care citește toate numerele din fișierul text **date.in** și afișează pe ecran lungimea maximă a unei secvențe din sir cu proprietatea că orice două elemente aflate pe poziții consecutive sunt prime între ele. O secvență a unui sir este formată din unul sau mai multe elemente aflate pe poziții consecutive.

Exemplu: dacă fișierul **date.in** are conținutul alăturat, se va afișa 3 pentru că cea mai lungă secvență cu proprietatea cerută este 6 25 6. (6p.)

7
16 25 6 12 10 4 5

53.

Subprogramul **dist**, cu doi parametri, primește prin intermediul primului parametru **a** un număr natural cu maximum 8 cifre și returnează prin intermediul celui de-al doilea parametru **b** numărul cifrelor distincte ale lui **a**.

Exemplu: **dist(1223712)** returnează valoarea 4 (deoarece cifrele distincte ale parametrului de apel sunt 1, 2, 3, 7)

a) Scrieți definiția completă a subprogramului **dist**. (4p.)

b) Fișierul text **date.in** conține pe prima linie un număr natural nenul **n** ($n \leq 100$), iar pe a doua linie **n** numere naturale, separate prin câte un spațiu, fiecare număr având maximum 8 cifre. Scrieți un program C/C++ care citește numerele din fișier și afișează pe ecran, despărțite prin câte un spațiu, numerele de pe a doua linie a fișierului text **date.in**, ce au număr maxim de cifre distincte, folosind apeluri utile ale subprogramului **dist**.

Exemplu: dacă fișierul **date.in** are conținutul | 6
alăturat, atunci se vor afișa numerele: 16 1775 333 242477 123 55566
1775 242477 123 (6p.)

54.

Se consideră subprogramul **pal** care primește care primește prin intermediul primului parametru **a** un număr natural, cu minimum 2 cifre și maximum 8 cifre, și returnează prin intermediul celui de-al doilea parametru **b** cel mai apropiat număr palindrom de **a**. În cazul în care există 2 astfel de numere, subprogramul va returna numărul mai mic. Un număr natural **x** este palindrom dacă este egal cu numărul obținut prin scrierea cifrelor lui **x** în ordine inversă.

Exemplu: **pal(16)** va returna valoarea 11; **pal(126)** va returna 121; **pal(33)** va returna 33

a) Scrieți definiția completă a subprogramului **pal**. (4p.)

b) Fișierul text **date.in** conține pe prima linie un număr natural nenul **n** ($n \leq 100$) iar pe a doua linie **n** numere naturale nenule, separate prin câte un spațiu, fiecare număr având minimum 2 cifre și maximum 8 cifre. Scrieți un program C/C++ care citește toate numerele din fișierul text **date.in** și afișează pe ecran despărțite printr-un spațiu, pentru fiecare dintre cele **n** numere cel mai apropiat număr palindrom, folosind apeluri utile ale subprogramului **pal**.

Exemplu: dacă fișierul **date.in** are conținutul alăturat, atunci | 4
se vor afișa numerele: 11 1771 333 191 (6p.) | 16 1775 333 190

55.

Se consideră subprogramul **cifre** care primește prin intermediul primului parametru **a** un număr natural cu maximum 8 cifre nenule și returnează, prin intermediul celui de-al doilea parametru **b**, cel mai mic număr care se poate forma cu toate cifrele distincte ale lui **a**.

a) Scrieți definiția completă a subprogramului **cifre**. (4p.)

b) Se consideră fișierul text **date.in** ce conține pe prima linie un număr natural nenul **n** ($n \leq 100$) iar pe a doua linie **n** numere naturale, separate prin spațiu, fiecare număr având maximum 8 cifre nenule. Scrieți un program C/C++ care citește toate numerele din fișierul text **date.in** și afișează pe ecran, despărțite printr-un spațiu, numerele situate pe a doua linie a fișierului, formate numai din cifre distincte ordonate strict crescător, folosind apeluri utile ale subprogramului **cifre**. În cazul în care nu există niciun astfel de număr se va afișa valoarea 0.

Exemplu: dacă fișierul **date.in** are conținutul alăturat, | 6
atunci se vor afișa numerele: 16 269 (6p.) | 16 175 333 242477 321 269

56.

Fisierul text **BAC.TXT** conține, pe o singură linie, cel puțin 3 și cel mult 100 de numere naturale nenule distincte de cel mult 4 cifre fiecare, numerele fiind separate prin câte un spațiu. Scrieți un program C/C++ care citește toate numerele din fișierul **BAC.TXT** și scrie pe ecran, în ordine descrescătoare, cele mai mici 3 numere citite.

Exemplu: dacă fișierul **BAC.TXT** conține numerele 1017 48 310 5710 162, atunci se va afișa: 310 162 48 (10p.)

57.

Fisierul text **INTRARE.TXT** conține, pe o singură linie, cel mult 100 de numere naturale nenule de cel mult patru cifre fiecare, numerele fiind separate prin câte un spațiu. Scrieți un program c/c++ care citește numerele din fișier și scrie în fișierul text **IESIRE.TXT** toate valorile obținute prin însumarea a câte două numere din fișierul **INTRARE.TXT**, ordonate crescător. Dacă o valoare se obține ca sumă a mai multor perechi de numere din fișierul **INTRARE.TXT**, ea va fi afișată o singură dată.

Exemplu:

INTRARE.TXT

1 4 3 2

IESIRE.TXT

3 4 5 6 7

(10p.)

58.

Fisierul text **bac.txt** contine, pe o singură linie, cel mult 100 de numere naturale nenule de cel mult 4 cifre fiecare, numerele fiind ordonate crescător și separate prin câte un spațiu. Scrieți un program c/c++ care citește de la tastatură un număr natural **x** de cel mult 4 cifre și verifică dacă **x** se află în fișierul **bac.txt**. În caz afirmativ, se va afișa pe ecran mesajul **DA**, altfel se va afișa mesajul **NU**.

Exemplu: dacă **x=312**, iar fișierul **bac.txt** conține numerele:

17 48 312 5742 8692

atunci se va afișa: **DA** ;

dacă **x=20**, iar fișierul **bac.txt** conține numerele:

17 48 312 5742 8692

atunci se va afișa: **NU** .

(10p.)

59.

Fisierul text **bac.txt** contine, pe o singură linie, cel puțin 3 și cel mult 100 de numere naturale nenule distincte de cel mult 4 cifre fiecare, numerele fiind separate prin câte un spațiu. Scrieți un program c/c++ care citește numerele din fișier și scrie pe ecran ultima cifră a produsului celor mai mari 3 numere citite.

Exemplu: dacă fișierul **bac.txt** conține numerele:

1017 48 312 5742 162

atunci se va afișa: 8 (ultima cifră a produsului numerelor 1017, 5742, 312) (10p.)

60.

Fisierul text **bac.txt** contine, pe o singură linie, cel mult 100 de numere naturale nenule de cel mult 4 cifre fiecare, numerele fiind ordonate crescător și separate prin câte un spațiu. Scrieți un program c/c++ care citește de la tastatură un număr natural **x** de cel mult 4 cifre și verifică dacă **x** se află în fișierul **bac.txt**. În caz afirmativ, se va afișa pe ecran mesajul **DA**, altfel se va afișa mesajul **NU**.

Exemplu: dacă **x=312**, iar fișierul **bac.txt** conține numerele:

17 48 312 5742 8692

atunci se va afișa: **DA** ;

dacă **x=20**, iar fișierul **bac.txt** conține numerele:

17 48 312 5742 8692

atunci se va afișa: **NU** .

(10p.)

61.

Subprogramul **cifra** primește prin singurul său parametru **x**, un număr real nenul pozitiv și furnizează prin parametrul **y** valoarea cifrei unităților părtii întregi a lui **x**.

Exemplu: la apelul **cifra(34.567)** se va returna 4.

a) Scrieți definiția completă a subprogramului **cifra**. (10p.)

b) Fișierul text **medii.txt** conține cel mult 600 de linii. Pe fiecare linie se află, separate printr-un spațiu, două numere reale, cu cel mult două zecimale, din intervalul **[1, 10]**, care reprezintă media pe semestrul 1 respectiv media pe semestrul al 2-lea, ale unui elev. În situațiile statistice pe care școala le realizează, fiecare medie este încadrată într-una dintre următoarele categorii de medii: **[3, 3.99]**, **[4, 4.99]**, **[5, 5.99]**, **[6, 6.99]**, **[7, 7.99]**, **[8, 8.99]**, **[9, 10]**. Scrieți un program C/C++ care citește datele din fișier și afișează pe ecran numărul elevilor care au media din semestrul al 2-lea în categoria imediat următoare categoriei căreia îl apartine media din semestrul 1. Ordinea categoriilor este cea din enumerarea de mai sus. În program se vor folosi apeluri utile ale subprogramului **cifra**. Se va utiliza un algoritm eficient din punctul de vedere al memoriei utilizate.

Exemplu: dacă fișierul **medii.txt** conține: Pe ecran se afișează:

9.45 7.90

2

6.34 7.60

8.75 9.90

(6p.)

c

62.

63.

Subprogramul **verif** primește prin singurul său parametru, **x**, un număr natural nenul cu cel mult 9 cifre și returnează valoarea 1 dacă numărul conține cel puțin o secvență de 3 cifre impare alăturate și 0 în caz contrar.

Exemplu: la apelul **verif(7325972)** se va returna valoarea 1.

a) Scrieți definiția completă a subprogramului **verif**. (10p.)

b) Fișierul text **date.txt** conține pe prima linie un număr natural nenul **n** cu cel mult 4 cifre și pe fiecare dintre următoarele **n** linii câte un număr natural, cu exact 6 cifre. Scrieți un program C/C++ care citește numerele din fișierul **date.txt** și afișează pe ecran, separate prin câte un spațiu, acele numere care au primele 3 cifre impare. Se vor utiliza apeluri utile ale subprogramului **verif**. Dacă nu există niciun număr cu această proprietate, se va afișa mesajul **nu**. Alegeți o metodă eficientă din punctul de vedere al memoriei utilizate.

De exemplu: dacă fișierul **date.txt** conține

3

133579

345796

973314

Pe ecran se afișează:

133579 973314

(6p.)

64.

Fișierul text `date.in` conține pe prima linie, separate prin câte un spațiu, cel mult 1000 numere naturale, fiecare dintre ele având maximum 9 cifre.

a) Scrieți un program c/c++ care citește numerele din fișierul `date.txt` și determină cea mai lungă secvență ordonată strict descrescător, formată din valori citite consecutiv din fișier. Numerele din secvență găsită vor fi afișate pe ecran, pe o linie, separate prin câte un spațiu. Dacă sunt mai multe secvențe care respectă condiția impusă, se va afișa doar prima dintre acestea. Alegeti o metodă de rezolvare eficientă din punctul de vedere al timpului de executare.

Exemplu: dacă fișierul `date.in` conține

5 2 9 4 3 6 3 2 1 0 8

pe ecran se afișează:

6 3 2 1 0

(6p.)

65.

Subprogram `sfx` primește prin singurul său parametru, `x`, un număr natural din intervalul $[100, 2000000000]$ și returnează valoarea 1 dacă ultimele trei cifre ale numărului sunt în ordine strict descrescătoare sau valoarea 0 în caz contrar.

Exemplu: la apelul `sfx(24973)` se va returna valoarea 1.

a) Scrieți definiția completă a subprogramului `sfx`. (10p.)

b) Fișierul text `date.in` conține cel mult 10000 de numere naturale de exact 6 cifre fiecare, separate prin câte un spațiu. Scrieți un program c/c++ care citește toate numerele din fișier, determină și afișează pe ecran câte dintre aceste numere au toate cifrele în ordine strict descrescătoare. Programul va folosi apele utile ale subprogramului `sfx`. Se va utiliza un algoritm eficient din punctul de vedere al memoriei utilizate. (6p.)

Exemplu: dacă fișierul `date.in` conține

236543

Pe ecran se afișează:

2

865210

976532

(6p.)

66.

Fișierele text `A.TXT` și `B.TXT` conțin cel mult 10000 de numere naturale cu cel mult 9 cifre fiecare, scrise fiecare pe câte o linie.

a) Scrieți un program c/c++ care citește numerele din cele două fișiere și, printr-o metodă eficientă din punct de vedere al timpului de executare și al spațiului de memorie utilizat și afișează pe ecran câte dintre numerele din fișierul `A.TXT` sunt strict mai mici decât toate numerele memorate în fișierul `B.TXT`. (6p.)

Exemplu: dacă fișierul `A.TXT` are conținutul alăturat

41111	iar fișierul <code>B.TXT</code> are conținutul	91111
81111	alăturat:	91111
11111		61111
91111		91111
51111		91111
111111		81111
31111		61111
431111		91111
61111		
201111		

atunci programul va afișa valoarea 4 deoarece 41111, 11111, 51111, 31111 sunt mai mici decât toate elementele din fișierul `B.TXT`

67.

Fișierul text **NUMERE.TXT** conține pe prima linie un număr natural n ($1 \leq n \leq 10000$) și pe a doua linie, un sir **crescător** de n numere naturale, fiecare având cel mult 9 cifre. Numerele de pe a doua linie sunt separate prin câte un spațiu.

a) Scrieți un program c/c++ care utilizând o metodă eficientă din punct de vedere al timpului de executare și al spațiului de memorie, afișează pe ecran elementele distincte ale sirului aflat pe a doua linie a fișierului. (6p.)

Exemplu: dacă fișierul **NUMERE.TXT** are

conținutul alăturat

111 111 111 2111 4111 71111 71111

atunci programul va afișa pe ecran 111 2111 4111 71111

68.

Fișierul text **SIR.TXT** conține pe prima linie un număr natural n ($1 \leq n \leq 10000$) și pe a doua linie, separate prin spații, un sir **crescător** de n numere naturale cu cel mult 9 cifre fiecare.

Numim platou într-un sir de valori, o secvență de elemente identice situate pe poziții alăturate. Lungimea unui platou este egală cu numărul de elemente care îl formează.

a) Scrieți un program c/c++ care citește valorile din fișier și, printr-o metodă eficientă din punct de vedere al timpului de executare și al spațiului de memorie utilizat afișează pe ecran, separate prin câte un spațiu, lungimea maximă a unui platou, precum și valoarea care formează platoul. În cazul în care sunt mai multe platouri de aceeași lungime se va afișa valoarea cea mai mare care formează unul dintre aceste platouri. (6p.)

Exemplu: dacă 10

fișierul **SIR.TXT** are 111 2111 2111 2111 3111 4111 4111 51111 51111 51111
conținutul alăturat

atunci programul va afișa pe ecran 3 51111

b) Descrieți succint, în limbaj natural, metoda utilizată la punctul a, justificând eficiența acesteia. (4p.)

69.

Fișierul text **NUMERE.TXT** conține pe prima linie un număr natural n ($1 \leq n \leq 10000$) și pe a doua linie, separate prin spații, n numere naturale cu cel mult 9 cifre fiecare. Aceste numere sunt dispuse în ordine **crescătoare** și separate între ele printr-un spațiu.

a) Scrieți un program c/c++ care citește valorile din fișier și, printr-o metodă eficientă din punct de vedere al timpului de executare și al spațiului de memorie utilizat, afișează pe ecran separate printr-un spațiu, în ordine crescătoare, numerele pare de pe a doua linie a fișierului, urmate de cele impare în ordine descrescătoare. (6p.)

Exemplu: dacă fișierul **NUMERE.TXT** are

conținutul alăturat

212 412 5111 71113 81112 101112

atunci programul va afișa pe ecran 212 412 81112 101112 71113 5111

70.

Fișierul text **NUMERE.TXT** conține pe prima linie un număr natural n ($1 \leq n \leq 10000$) și pe a doua linie, n numere naturale cu cel mult 9 cifre fiecare, numere nu neapărat distincte. Aceste numere sunt dispuse în ordine **crescătoare** și separate între ele printr-un spațiu.

a) Scrieți un program c/c++ care citește valorile din fișier și, printr-o metodă eficientă din punct de vedere al timpului de executare și al spațiului de memorie utilizat, afișează pe ecran, cu câte un spațiu între ele, valoarea care apare de cele mai multe ori în fișier și de câte ori apare ea. Dacă există mai multe valori care apar de un număr maxim de ori, se va afișa cea mai mică dintre ele. (6p.)

Exemplu: dacă fișierul

NUMERE.TXT are conținutul 711 711 711 11111 11111 11111 191111 231111
alăturat

atunci programul va afișa pe ecran 711 3