4. Se consideră fişierul **BAC.TXT** ce conține un şir **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 şi al timpului de executare, citeşte din fişier toți termenii şirului şi afişează pe ecran, pe o singură linie, fiecare termen distinct al şirului urmat de numărul de apariții ale acestuia în şir. Valorile afişate sunt separate prin câte un spațiu.

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

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

programul va afişa:

```
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.)**

6

4. Subprogramul **sub** primeşte prin intermediul parametrilor:
– **n** şi **m** două numere naturale (**1<n<100, 1<m<100**)
– **a** şi **b** două tablouri unidimensionale, fiecare având componente numere naturale de maximum patru cifre, **ordonate crescător**; tabloul **a** conține **n** numere pare, iar tabloul **b** conține **m** numere impare.
Subprogramul va afişa pe ecran, în ordine crescătoare, separate prin câte un spațiu, un şir format dintr-un număr maxim de elemente care aparțin cel puțin unuia dintre tablouri, astfel încât orice două elemente aflate pe poziții consecutive să fie de paritate diferită.
Exemplu: pentru **n=5**, **m=3** şi tablourile **a=(2,4,8,10,14)** şi **b=(3,5,11)**, subprogramul va afişa **2 3 4 5 8 11 14** sau **2 3 4 5 10 11 14.**

a) Scrieți definiția completă a subprogramului **sub**, alegând pentru rezolvare un algoritm eficient din punctul de vedere al timpului de executare. **(6p.)**

b) Descrieți succint, în limbaj natural, algoritmul pe baza căruia a fost scris subprogramul de la punctul **a)**, explicând în ce constă eficiența metodei utilizate. **(4p.)**

8

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

a) Scrieți un program C/C++ care citeşte toate numerele din fişierul **BAC.TXT** şi 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 fişier. Cele două numere vor fi afişate pe ecran în ordine descrescătoare, cu un spațiu între ele. Dacă nu pot fi determinate două astfel de numere, programul va afişa pe ecran valoarea 0.
Exemplu: dacă fişierul **BAC.TXT** conține numerele:

```
12 2345 123 67 989 6 999 123 67 989 999
```

atunci programul va afişa

```
998 997
```

(6p.)

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

9

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

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 momentul vânzării respective.

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 perechile următoare, nu neapărat în această ordine:

```
1 150
2 30
3 5
```

(6p.)

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

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

10

3. Fișierul text **numere.txt** conține pe prima linie un număr natural **n** (**n<30000**), iar pe a doua linie **n** numere întregi având maximum **4** cifre fiecare. Se cere să se afișeze pe ecran un șir de **n** numere întregi, cu proprietatea că valoarea termenului de pe poziția **i** (**i=1,2,…,n**) din acest șir 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 8 9 10 10
```

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

11

3. 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.)**

12

3. Se consideră şirul `1, 2,1, 3,2,1, 4,3,2,1, ...`
construit astfel: prima grupă este formată din numărul `1`, a doua grupă este formată din numerele `2` şi `1`, iar grupa a `k`-a, este formată din numerele `k`, `k-1`,..., `1`.
Se cere să se citească de la tastatură un număr natural `n` (`n≤1000`) şi să se afişeze pe ecran cel de al `n`-lea termen al şirului dat.

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

b) Scrieți programul `c/c++` corespunzător algoritmului descris **(6p.)** 13

3. Se citeşte de la tastatură un număr natural `n` (`n≤500`) şi apoi `n` cifre separate prin spații. Se cere să se afişeze pe ecran cele `n` cifre citite, în ordine crescătoare, separate prin câte un spațiu.
Exemplu: pentru `n=19` şi cifrele `3 3 0 9 2 1 2 1 3 7 1 5 2 7 1 0 3 2 3` se va afişa pe ecran `0 0 1 1 1 1 2 2 2 2 3 3 3 3 3 5 7 7 9`.

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

b) Scrieți programul `c/c++` corespunzător algoritmului descris. **(6p.)** 14.

4. În fişierul 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 şi să se afişeze pe ecran, separate printr-un spațiu, ultimele **două** numere impare (nu neapărat distincte) din fişierul `BAC.IN`. Dacă în fişier se găseşte un singur număr impar sau niciun număr impar se va scrie pe ecran mesajul `Numere insuficiente`.

Exemplu: dacă fişierul `BAC.IN` conține valorile: `12 15 68 13 17 90 31 42` se va afişa `17 31`.

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

b) Scrieți programul `c/c++` corespunzător algoritmului descris. **(6p.)** 15

4. În fişierul `numere.txt` sunt memorate maximum `10000` de numere naturale cu cel mult `9` cifre fiecare. Fiecare linie a fişierului conține câte un număr. Se cere afişarea pe ecran, în ordine descrescătoare, a tuturor cifrelor care apar în numerele din fişier. Alegeți un algoritm de rezolvare eficient din punct de vedere al timpului de executare.
Exemplu: dacă fişierul `numere.txt` conține:
`267`
`39628`
`79`
se va tipări `9987766322`.

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

b) Scrieți programul `c/c++` corespunzător algoritmului ales. **(6p.)** 16

4. În fişierul **numere.txt** pe prima linie este memorat un număr natural **n** (**n≤10000**), iar pe linia următoare un şir de **n** numere naturale distincte două câte două, separate prin câte un spațiu, cu maximum **4** cifre fiecare. Se cere afişarea pe ecran a poziției pe care s-ar găsi primul element din şirul aflat pe linia a doua a fişierului, în cazul în care şirul ar fi ordonat crescător. Numerotarea pozițiilor elementelor în cadrul şirului este de la **1** la **n**. Alegeți un algoritm de rezolvare eficient din punct de vedere al memoriei utilizate şi al timpului de executare.

Exemplu: dacă fişierul **numere.txt** conține:

```
6
267 13 45 628 7 79
```

se va afişa **5**, deoarece primul element din şirul inițial, **267**, s-ar găsi pe poziția a cincea în şirul ordonat crescător (**7 13 45 79 267 628**).

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

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

17

4. În fişierul **numere.txt** este memorat un şir 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 afişarea pe ecran a poziției pe care se va găsi acesta în şirul de numere din fişier, dacă şirul ar fi ordonat descrescător, sau mesajul **nu există**, dacă numărul **k** nu se află printre numerele din fişier. Alegeți un algoritm eficient de rezolvare din punct de vedere al memoriei utilizate şi al timpului de executare.

Exemplu: dacă fişierul **numere.txt** conține numerele **26 2 5 30 13 45 62 7 79**, iar **k** are valoarea **13**, se va afişa **6** deoarece **13** s-ar găsi pe poziția a şasea în şirul ordonat descrescător (**79 62 45 30 26 13 7 5 2**).

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

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

18

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

Exemplu: pentru următoarele fişiere:

`nr1.txt`
```
5
3 6 8 9 12
```
`nr2.txt`
```
6
2 3 5 7 9 13
```
se va afişa `2 3 5 6 7 8 9 12 13`.

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

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

19

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

Exemplu: pentru următoarele fişiere:

`nr1.txt`
```
5
3 6 8 9 12
```
`nr2.txt`
```
6
2 3 5 7 9 13
```
se va afişa `3 9`.

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

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

20

4. Fişierul text **BAC.TXT** conţine pe prima linie două numere naturale **n** şi **k** separate de un spaţiu ($3 \le n \le 10000$, $2 \le k \le n/2$), iar pe a doua linie un şir de **n** numere naturale $x_1, x_2, \ldots, x_n$ separate prin câte un spaţiu, fiecare număr din acest şir având cel mult patru cifre.

a) Scrieţi un program C/C++ care citeşte numerele din fişier şi determină, utilizând o metodă eficientă din punct de vedere al timpului de executare, cel mai mic indice **i** ($1 \le i \le n-k+1$) pentru care media aritmetică a numerelor $x_i, x_{i+1}, \ldots, x_{i+k-1}$ este maximă. Programul afişează valoarea lui **i** pe ecran.
Exemplu: pentru fişierul alăturat se afişează 2, deoarece media maximă se obţine pentru 9, 4, 7. **(6p.)**

```
8 3
2 9 4 7 5 2 9 9
```

b) Explicaţi succint, în limbaj natural, metoda utilizată la punctul **a**, justificând eficienţa acesteia. **(4p.)**

21

4. Fişierul text **bac.txt** conţine pe prima linie numărul natural **n**, $1 \le n \le 30000$, pe următoarele **n** linii un şir de **n** numere întregi, ordonate crescător, iar pe ultima linie două numere întregi **a** şi **b** ($a \le b$) separate de un spaţiu. Fiecare dintre cele **n** numere, precum şi valorile **a** şi **b**, au cel mult patru cifre.

a) Scrieţi un program C/C++, eficient din punct de vedere al timpului de executare, care afişează pe ecran cel mai mic număr întreg din intervalul închis **[a,b]** care se găseşte în şirul dat. Dacă nu există un astfel de număr, programul afişează textul **NU**.
Exemplu: dacă fişierul **bac.txt** are conţinutul alăturat, programul afişează valoarea 11 **(6p.)**

b) Descrieţi în limbaj natural metoda utilizată şi explicaţi în ce constă eficienţa ei. **(4p.)**

```
4
-2
7
11
35
8 15
```

24

4. Fişierul text **NUMAR.TXT** conţine pe prima linie un număr real pozitiv **x** care are cel mult **două** cifre la partea întreagă şi cel mult **şapte** cifre după punctul zecimal..

a) Scrieţi un program C/C++ care, utilizând un algoritm eficient din punct de vedere al timpului de executare şi al memoriei utilizate, afişează pe ecran, separate printr-un spaţiu, două numere naturale al căror raport este egal cu **x** şi a căror diferenţă absolută este minimă.
Exemplu: dacă fişierul conţine valoarea alăturată, se vor afişa pe ecran numerele 3 8. **(6p.)**

```
0.375
```

b) Descrieţi în limbaj natural metoda utilizată şi explicaţi în ce constă eficienţa ei. **(4p.)**

25

4. Fişierul text **NUMERE.IN** conține pe prima linie un număr natural nenul **n** ($2 \leq n \leq 100$) şi 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 şi afişează pe ecran cel mai mare număr natural **x** cu proprietatea că în orice interval deschis având drept capete oricare două dintre cele **n** numere aflate pe linia a doua în fişierul **NUMERE.IN** se găsesc cel puțin **x** numere întregi.
Exemplu: dacă fişierul **NUMERE.IN** are conținutul:
6
3.5 5.1 9.2 16 20.33 100
atunci se afişează 2

Explicație: 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.

b) Descrieți în limbaj natural metoda utilizată şi explicați în ce constă eficiența ei. **(4p.)** 27

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

De exemplu tabloul

A	7	3	4	1	6	4	6	9	7	1	8	7

se poate reduce la tabloul

B	14	7	26	16

Fişierul text **NUMERE.IN** conține pe prima linie două numere naturale nenule **n** şi **m** ($1 \leq m \leq n \leq 100$), pe linia a doua **n** numere naturale din intervalul [1;10000] şi 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 citeşte toate numerele din fişierul **NUMERE.IN** şi verifică, utilizând un algoritm eficient din punctul de vedere al timpului de executare, dacă tabloul construit cu cele **n** numere aflate pe linia a doua în fişier se poate reduce la tabloul construit cu cele **m** numere aflate pe linia a treia în fişier. Programul afişează pe ecran mesajul **DA** în caz afirmativ şi mesajul **NU** în caz negativ. **(6p.)**

b) Descrieți în limbaj natural metoda utilizată şi explicați în ce constă eficiența ei. **(4p.)** 29

4. Fişierul text **NUMERE.IN** conține pe prima linie un număr natural nenul **n** **(1≤n≤100)** şi pe următoarea linie **n** numere reale pozitive **ordonate crescător**, separate prin câte un spațiu.

a) Scrieți un program **C/C++** care citeşte din fişierul **NUMERE.IN** numărul natural **n**, şi determină, utilizând un algoritm eficient din punct de vedere al timpului de executare şi 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 fişier.

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

```
6
2.3  2.3  2.8  5.7  5.7  6.3
```

atunci se afişează **3** (intervalele **[2,3]**, **[5,6]**, **[6,7]** sunt cele **3** intervale de forma cerută care conțin numere din şir). **(6p.)**

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

30

4. În fişierul **numere.txt** se află memorate, pe prima linie un număr natural **n** (**1≤n≤100**), iar pe fiecare dintre următoarele **n** linii, câte două numere întregi **x,y** (**-100≤x≤y≤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 şi al spațiului de memorare, care citeşte din fişier datele existente, determină segmentul rezultat în urma intersecției tuturor celor **n** segmente date şi afişează pe ecran două numere despărțite printr-un spațiu ce reprezintă capetele segmentului cerut. Dacă segmentele nu au nici un punct comun se va afişa pe ecran valoarea **0**. **(6p.)**

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

Exemplu: dacă fişierul **numere.txt** are conținutul alăturat, se va afişa pe ecran

```
3 5
```

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

31

4. În fişierul **numere.txt** sunt memorate pe mai multe linii, numere întregi (cel mult **100**), numerele de pe aceeaşi 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 fişier şi să se afişeze pe ecran aceste valori, despărțite printr-un spațiu. Dacă în fişier nu se află două astfel de valori, pe ecran se va afişa valoarea **0**.

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

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

Exemplu: dacă fişierul **numere.txt** are conținutul alăturat, se va afişa pe ecran, nu neapărat în această ordine:

```
-77 10
```

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

32

4. Pe prima linie a fişierului **numere.txt** se află un număr natural **n** (**n≤100**), iar pe următoarele **n** linii, câte **n** numere naturale despărțite prin câte un spațiu, fiecare având cel mult **9** cifre. Printre aceste numere se află cel puțin unul cu **3** cifre şi cel puțin unul cu **4** cifre.

a) Scrieți în limbajul **C/C++**, un algoritm eficient din punct de vedere al gestionării memoriei care citeşte din fişier datele existente şi determină şi afişează pe ecran, separate printr-un spațiu, două numere din fişier, **x** şi **y**, unde **x** este cel mai mare număr de trei cifre, iar **y** este acel număr pentru care **|x-y|** are valoare minimă. Dacă sunt mai multe valori pentru **y** care respectă condiția impusă se va afişa numai una dintre ele. **(10p.)**

b) Explicați în limbaj natural metoda utilizată justificând eficiența acesteia. **(4p.)**

Exemplu: dacă fişierul **numere.txt** are conținutul alăturat, se va afişa:

```
800 1100
```

```
5
112  333  1 500  1100
 1   95  7 97 12
45 800  0  7   89
1   5  17 197 102
45 86  0  7   9
```

33

3. **a)** Subprogramul **max** primeşte ca parametru un tablou unidimensional **x** cu cel mult **100** de elemente numere întregi, care sunt, în ordine, termenii unei progresii aritmetice şi un număr natural **n**, care reprezintă dimensiunea tabloului. Scrieți definiția completă a subprogramului **max** care returnează cel mai mare termen al progresiei aritmetice. Alegeți un algoritm de rezolvare eficient din punct de vedere al timpului de executare. **(6p.)**

b) Explicați în limbaj natural metoda utilizată justificând eficiența acesteia. **(4p.)**

c) Pe prima linie a fişierului **numere.txt** se află un număr natural **n** (**n≤100**), iar pe următoarele **n** linii, câte **n** numere întregi cu cel mult **4** cifre fiecare. Scrieți programul **C/C++** care citeşte din fişier datele existente, determină liniile din fişier pe care s-au memorat în ordine termenii unei progresii aritmetice şi afişează pe ecran, folosind apeluri ale subprogramului **max** cel mai mare număr (diferit de cel situat pe prima linie) din fişier, care în plus este termenul unei progresii aritmetice. **(10p.)**

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

```
(-9 -7  -5  -3  -1),
(50 40 30 20 10) şi
(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
```

34

3. Subprogramul `cifra` primeşte prin intermediul parametrului `a` un număr natural cu cel mult `4` cifre şi returnează ultima cifră pară a sa. Dacă numărul nu conține cifre pare, subprogramul returnează valoarea `-1`. De exemplu, dacă `a=8345`, subprogramul va returna `4`.

a) Să se scrie definiția completă a subprogramului `cifra`. **(10p.)**

b) Pe prima linie a fişierului `bac.in` se află un număr natural nenul `n` `(n≤15000)`, iar pe a doua linie a fişierului se află un şir de `n` numere naturale, despărțite prin câte un spațiu, fiecare număr fiind format din cel mult `4` cifre.
Scrieți un program `C/C++` care citeşte numerele din fişier şi afişează pe ecran, folosind apeluri utile ale subprogramului `cifra`, cel mai mare număr care se poate forma cu ultimele cifre pare ale fiecărui element, dacă acestea există. Alegeți o metodă de rezolvare eficientă ca timp de executare. Dacă toate numerele de pe a doua linie a fişierului au numai cifre impare, programul va afişa mesajul `NU EXISTA`.

Exemplu: dacă fişierul `bac.in` are conținutul alăturat, pe ecran se va afişa: `64220` **(6p.)**

```
7
369 113 2 0 33 1354  42
```

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

36

3. **a)** Pe prima linie a fişierului `bac.in` se află un număr natural nenul `n` `(n≤1000)`, iar pe a doua linie a fişierului se află un şir format din `n` numere naturale, despărțite prin câte un spațiu, fiecare număr fiind format din cel mult `4` cifre. Scrieți un program `C/C++` care citeşte numerele din fişier şi care afişează pe ecran mesajul `DA` dacă elementele pare în şir sunt în ordine crescătoare, iar cele impare sunt în ordine descrescătoare şi mesajul `NU` în caz contrar. Alegeți un algoritm eficient ca timp de executare şi spațiu de memorie utilizat. **(6p.)**

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

Exemplu: dacă fişierul `bac.in` are conținutul alăturat, pe ecran se va afişa: `DA`

```
8
10 1133 12 331 42 1354 221 13
```

40

4. Fişierul text `numere.txt` conține pe prima linie un număr natural `n` (`0<n<100000`), iar pe a doua linie un şir format din `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, cea mai mare cifră dintre cele situate pe a doua linie a fişierului, precum şi numărul de apariții ale acesteia. Cele două numere vor fi afişate pe o singură linie a ecranului, separate printr-un spațiu.

Exemplu: daca fişierul `numere.txt` are următorul conținut:

```
7
3 5 2 1 5 3 1
```

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

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

41

4. Fişierul 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 două 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 fişierului. Aceste numere vor fi afişate pe ecran în ordine crescătoare, separate prin câte un spațiu.

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

```
7
3 5 2 1 5 23 1
```

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

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

42

4. Fişierul 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 fişierului **numere.txt**. Numărul determinat se va afişa pe ecran.

Exemplu: daca fişierul **numere.txt** are următorul conținut:

```
7
2 5 3 1 5 8 9
```

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

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

43

4. Fişierul 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 fişierului. Programul va afişa pe ecran aceste cifre in ordine crescătoare, separate prin câte un spațiu.

Exemplu: daca fişierul **numere.txt** are următorul conținut:

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

atunci pe ecran se va afişa: **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.)**

44

4. Fişierul 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 două cifre, separate prin câte un spațiu.

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

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

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

atunci pe ecran se va afişa: **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.)**

45

4. **a)** Fişierul **date.in** conține un şir de cel mult **10000** numere naturale (printre care cel puțin un număr par şi cel puțin un număr impar), cu cel mult **2** cifre fiecare, separate prin câte un spațiu. Scrieți un program **C/C++** care citeşte numerele din fişierul **date.in** şi scrie în fişierul text **date.out** valorile distincte citite, separate prin câte un spațiu, respectându-se regula: pe prima linie vor fi scrise numerele impare în ordine crescătoare, iar pe linia a doua numerele pare, în ordine descrescătoare. Alegeți o metodă eficientă din punctul de vedere al timpului de executare. **(6p.)**

Exemplu: dacă pe prima linie a fişierului **date.in** se află numerele:

```
75 12 3 3 18 75 1 3
```

atunci fişierul **date.out** va conține:

```
1 3 75
18 12
```

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

62

4. Pentru un şir de numere naturale, numim "nod" al şirului un termen din şir care are doi vecini, termenul precedent şi termenul următor din şir, şi valoarea termenului respectiv este strict mai mică decât suma valorilor celor doi vecini ai săi.

a) Fişierul text **date.in** conține un şir de cel puțin două şi cel mult **10000** de numere naturale având maximum **6** cifre fiecare, numere separate prin câte un spațiu. Scrieți un program **C/C++** care citeşte toate numerele din fişier şi afişează numărul de "noduri" ale şirului citit, folosind un algoritm eficient din punctul de vedere al memoriei utilizate. **(6p.)**

Exemplu: dacă fişierul **date.in** are următorul conținut:

```
51  20  100  43  43  618  5000  31  2020  114  116  4
```

atunci pe ecran se afişează **6** (cele şase numere subliniate reprezintă "noduri" ai şirului)

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

63

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

a) Scrieți un program **C/C++** care citeşte numerele din fişierul **date.txt** şi determină cea mai lungă secvență ordonată strict descrescător, formată din valori citite consecutiv din fişier. Numerele din secvența găsită vor fi afişate pe ecran, pe o linie, separate prin câte un spațiu. Dacă sunt mai multe secvențe care respectă condiția impusă, se va afişa doar prima dintre acestea. Alegeți o metodă de rezolvare eficientă din punctul de vedere al timpului de executare.

Exemplu: dacă fişierul **date.in** conține	pe ecran se afişează:
`5 2 19 4 3 6 3 2 1 0 8`	`6 3 2 1 0`

(6p.)

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

64

4. Pentru un şir de numere naturale, numim "pol" al şirului un termen din şir care are doi vecini, termenul precedent şi termenul următor din şir, şi valoarea termenului respectiv este strict mai mare decât valoarea fiecăruia dintre cei doi vecini ai săi.

a) Fişierul text **date.in** conține un şir de cel mult **10000** de numere naturale având maximum **6** cifre fiecare, numere separate prin câte un spațiu. Scrieți un program **C/C++** care citeşte toate numerele din fişier şi afişează numărul de "poli" ai şirului citit, folosind un algoritm eficient din punctul de vedere al memoriei utilizate. **(6p.)**

Exemplu: dacă fişierul **date.in** are următorul conținut:

`51 20 100 43 43 618 5000 31 2020 114 116 4`

atunci pe ecran se afişează **4** (cele patru numere subliniate reprezintă "poli" ai şirului)

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

65

4. Fişierele text **A.TXT** şi **B.TXT** conţin cel mult **10000** de numere naturale cu cel mult **9** cifre fiecare, scrise fiecare pe câte o linie.

a) Scrieţi un program **C/C++** care citeşte numerele din cele două fişiere şi, printr-o metodă eficientă din punct de vedere al timpului de executare şi al spaţiului de memorie utilizat, afişează pe ecran câte dintre numerele din fişierul **A.TXT** sunt strict mai mici decât toate numerele memorate în fişierul **B.TXT**. **(6p.)**

Exemplu: dacă fişierul **A.TXT** are conţinutul alăturat,

```
41111
81111
11111
91111
51111
111111
31111
431111
61111
201111
```

iar fişierul **B.TXT** are conţinutul alăturat:

```
91111
91111
61111
91111
91111
81111
61111
91111
```

atunci programul va afişa valoarea **4**, deoarece **41111**, **11111**, **51111**, **31111** sunt mai mici decât toate elementele din fişierul **B.TXT**.

b) Descrieţi succint, în limbaj natural, metoda utilizată la punctul **a**, justificând eficienţa acesteia. **(4p.)**

66

4. Fişierul text **NUMERE.TXT** conţine pe prima linie un număr natural **n** **(1≤n≤10000)** şi pe a doua linie un şir **crescător** de **n** numere naturale, fiecare având cel mult **9** cifre. Numerele de pe a doua linie sunt separate prin câte un spaţiu.

a) Scrieţi un program **C/C++** care, utilizând o metodă eficientă din punct de vedere al timpului de executare şi al spaţiului de memorie, afişează pe ecran elementele distincte ale şirului aflat pe a doua linie a fişierului. **(6p.)**

Exemplu: dacă fişierul **NUMERE.TXT** are conţinutul alăturat

```
7
111 111 111 2111 4111 71111 71111
```

atunci programul va afişa pe ecran **111 2111 4111 71111**.

b) Descrieţi succint, în limbaj natural, metoda utilizată la punctul **a)**, justificând eficienţa acesteia. **(4p.)**

67

4. Fişierul text **SIR.TXT** conține pe prima linie un număr natural **n** (1≤n≤10000) şi pe a doua linie, separate prin spații, un şir **crescător** de **n** numere naturale cu cel mult 9 cifre fiecare.
Numim platou într-un şir 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 citeşte valorile din fişier şi, printr-o metodă eficientă din punct de vedere al timpului de executare şi al spațiului de memorie utilizat afişează pe ecran, separate printr-un spațiu, lungimea maximă a unui platou, precum şi valoarea care formează platoul. În cazul în care sunt mai multe platouri de aceeaşi lungime se va afişa valoarea cea mai mare care formează unul dintre aceste platouri. **(6p.)**

Exemplu: dacă fişierul **SIR.TXT** are conținutul alăturat,

```
10
11 211 211 211 328 400 400 1201 1201 1201
```

atunci programul va afişa pe ecran `3 1201`.

b) Descrieți succint, în limbaj natural, metoda utilizată la punctul **a**, justificând eficiența acesteia. **(4p.)** 68

4. Fişierul text **NUMERE.TXT** conține pe prima linie un număr natural **n** (1≤n≤10000) şi pe a doua linie, **n** numere naturale cu cel mult 9 cifre fiecare. Aceste numere sunt dispuse în ordine **crescătoare** şi separate între ele prin câte un spațiu.

a) Scrieți un program C/C++ care citeşte valorile din fişier şi, printr-o metodă eficientă din punct de vedere al timpului de executare, afişează pe ecran, separate prin câte un spațiu, în ordine crescătoare, numerele pare de pe a doua linie a fişierului, urmate de cele impare în ordine descrescătoare. **(6p.)**

Exemplu: dacă fişierul **NUMERE.TXT** are conținutul alăturat

```
6
212 412 5111 71113 81112 101112
```

atunci programul va afişa pe ecran `212 412 81112 101112 71113 5111`

b) Descrieți succint, în limbaj natural, metoda utilizată la punctul **a**, justificând eficiența acesteia. **(4p.)** 69

4. Fişierul text **NUMERE.TXT** conține pe prima linie un număr natural **n** (1≤n≤10000) şi pe a doua linie, **n** numere naturale cu cel mult 9 cifre fiecare, numere nu neapărat distincte. Aceste numere sunt dispuse în ordine **crescătoare** şi separate între ele prin câte un spațiu.

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

Exemplu: dacă fişierul **NUMERE.TXT** are conținutul alăturat,

```
8
711 711 711 11111 11111 11111 191111 231111
```

atunci programul va afişa pe ecran `711 3`.

b) Descrieți succint, în limbaj natural, metoda utilizată la punctul **a**, justificând eficiența acesteia. **(4p.)** 70

3. Un număr natural se numeşte palindrom dacă numărul citit de la stânga la dreapta este egal cu numărul citit de la dreapta la stânga.

a) Scrieți definiția completă a subprogramului `Palindrom` care primeşte prin intermediul parametrului `n` un număr natural de cel mult nouă cifre şi returnează `1` dacă acesta este palindrom şi `0` în caz contrar. **(10p.)**

b) Fişierul text **NUMERE.IN** conține cel mult `100000` numere naturale de cel mult nouă cifre fiecare, numerele fiind despărțite prin câte un spațiu. Cel puțin unul dintre numere este palindrom.

Scrieți programul C/C++ care citeşte numerele din fişierul **NUMERE.IN** şi, folosind apeluri utile ale subprogramului `Palindrom` determină în mod eficient, din punct de vedere al memoriei utilizate şi al timpului de executare, care este cel mai mare număr palindrom citit şi de câte ori apare el în fişierul **NUMERE.IN**. Programul scrie în fişierul text **NUMERE.OUT** numărul astfel determinat precum şi numărul de apariții ale acestuia, pe rânduri diferite.
Exemplu: dacă **NUMERE.IN** conține numerele:
`23 565 78687 7887 7865 78687 7887 23 78687 98798`
atunci **NUMERE.OUT** va conține:
`78687`
`3` **(6p.)**

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

71

3. **a)** Scrieți definiția completă a subprogramului `Ecuatie` care primeşte prin parametrii `a`, `b` şi `c` trei numere întregi, $a \neq 0$, de cel mult patru cifre fiecare, reprezentând coeficienții ecuației de gradul al II-lea: $ax^2+bx+c=0$. În funcție de soluțiile ecuației subprogramul va returna:
- cea mai mare dintre soluții dacă ecuația are două soluții reale distincte, dintre care cel puțin una pozitivă.
- una dintre soluții dacă ecuația are două soluții egale şi pozitive.
- `-32000` în celelalte cazuri. **(10p.)**

b) Se consideră şirul `s: 1, 1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 6, 1, 2,...` .
Pentru un număr natural `k`, $0<k\leq 10000$, se cere să se determine valoarea elementului ce se află pe poziția `k` în şirul `s`.
Exemplu: pentru `k=5` numărul cerut este `2`.

Scrieți un program C/C++ care citeşte de la tastatură valoarea numărului natural `k` şi, prin apeluri utile ale funcției `Ecuatie`, determină valoarea elementului ce se află pe poziția `k` în şirul `s`, folosind un algoritm eficient din punctul de vedere al spațiului de memorie alocat şi al timpului de executare. Valoarea astfel determinată se va scrie în fişierul text `sir.out`. **(6p.)**

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

72

3. **a)** Scrieți definiția completă a subprogramului **Cautare**, cu trei parametri, **n**, **x** și **v**, care primește prin parametrul **n** un număr natural (**1≤n≤1000**), prin parametrul **x** un tablou unidimensional format din **n** componente (numere întregi de cel mult patru cifre fiecare: x_1, x_2, ..., x_n) memorate în ordine crescătoare și prin parametrul **v** un număr întreg de cel mult patru cifre, diferit de oricare dintre elementele tabloului unidimensional **x**.
Subprogramul va căuta, în mod eficient din punct de vedere al timpului de executare, poziția pe care ar trebui inserată valoarea **v** în șirul **x** astfel încât să se obțină tot un șir ordonat și returnează această poziție. **(6p.)**

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

c) Fișierul text **sir.in** conține cel mult **1000** numere naturale de maximum patru cifre fiecare, numerele fiind diferite două câte două și despărțite prin câte un spațiu.
Scrieți un program **C/C++** care citește numerele din fișierul **sir.in** și, folosind apeluri utile ale subprogramului **Cautare**, construiește în memorie un tablou unidimensional care va conține toate numerele din fișierul **sir.in** ordonate crescător. Programul scrie în fișierul text **sir.out** șirul obținut, câte **10** elemente pe un rând, elementele de pe același rând fiind despărțite printr-un singur spațiu.
Exemplu: dacă fișierul **sir.in** conține numerele: **7 -5 635 -456 0 8 587 -98 65 3 -8**
atunci după executarea programului fișierul **sir.out** va conține:
-456 -98 -8 -5 0 3 7 8 65 587
635 **(10p.)**

73

4. Se numește "număr mare" orice număr natural care are mai mult de nouă cifre.

a) Scrieți un program **C/C++** care citește de pe prima linie a fișierului text **NUMERE.IN** un număr natural **n** (**10<n<1000**), iar de pe a doua linie **n** cifre despărțite prin câte un spațiu, dintre care cel puțin una nenulă, și afișează pe ecran cel mai mic "număr mare" format cu toate cele **n** cifre din fișier. Alegeți o metodă eficientă din punct de vedere al utilizării memoriei. **(6p.)**

b) Descrieți succint în limbaj natural metoda de rezolvare folosită explicând în ce constă eficiența ei (3 – 4 rânduri). **(4p.)**
Exemplu: dacă fișierul **NUMERE.IN** conține
10
7 9 4 0 9 0 1 1 8 8
atunci se va afișa pe ecran "numărul mare" :
1001478899

74

3. **a)** Scrieți definiția completă a funcției `UltimaCifra` care primește prin cei doi parametri `a` și `b` câte un număr natural (`0<a<1000000`, `0<b<1000000`), calculează în mod eficient din punct de vedere al timpului de executare și returnează ultima cifră a numărului a^b (`a` la puterea `b`). **(6p.)**

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

c) Fișierul text `SIR.IN` conține pe prima sa linie un număr natural `n` (`0<n<1001`), iar pe fiecare dintre următoarele `n` linii câte o pereche de numere naturale, x_i y_i ($1 \le i \le n$, $x_i \le 30000$, $y_i \le 30000$).
Scrieți programul C/C++ care citește numerele din fișierul `SIR.IN` și scrie în fișierul text `SIR.OUT` ultima cifră expresiei: $X_1^{y_1} + X_2^{y_2} + \ldots + X_n^{y_n}$, folosind apeluri ale funcției `UltimaCifra`.

Exemplu: dacă fișierul `SIR.IN` are conținutul alăturat, atunci `SIR.OUT` va conține cifra `0`. **(10p.)**

```
3
25 6
8 10
1 4589
```

75

4. Fișierul `BAC.TXT` are pe prima linie două numere naturale `n` și `m` (`0<n<1000`, `0<m<1000`) separate prin câte un spațiu, pe linia a doua `n` numere întregi ordonate strict crescător, iar pe linia a treia `m` numere naturale distincte. Numerele din fișier aflate pe linia a doua și a treia au cel mult 6 cifre fiecare și sunt despărțite în cadrul liniei prin câte un spațiu. Să se scrie un program care citește toate numerele din fișier și afișează pe ecran, despărțite prin câte un spațiu, toate numerele de pe a doua linie a fișierului care apar cel puțin o dată și pe linia a treia a acestuia.

Exemplu: dacă fișierul are următorul conținut:

```
6 5
2 3 4 5 8 9
4 5 2 11 8
```

atunci se va afișa: `5 2 8`, nu neapărat în această ordine.

a) Descrieți în limbaj natural o metodă de rezolvare eficientă ca timp de executare. **(4p.)**

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

76

4. Fişierul text **bac.in** conține pe prima sa linie un număr natural `n` (`0<n<10000`), iar pe următoarea linie `n` numere naturale din intervalul `[1,100]` separate prin câte un spațiu. Se cere să se citească din fişier toate numerele şi să se afişeze pe ecran numărul sau numerele care apar de cele mai multe ori printre numerele citite de pe a doua linie a fişierului. Numerele afişate vor fi separate prin câte un spațiu. Alegeți un algoritm de rezolvare eficient atât din punctul de vedere al timpului de executare cât şi al gestionării memoriei. .

Exemplu: dacă fişierul **bac.in** are următorul conținut:

```
12
1 2 2 3 2 9 3 3 9 9 7 1
```

pe ecran se vor afişa valorile `2`, `3` şi `9`, nu neapărat în această ordine.

a) Explicați în limbaj natural metoda utilizată justificând eficiența acesteia (4-6 rânduri) **(4p.)**

b) Scrieți programul `c/c++` ce rezolvă problema enunțată, corespunzător metodei descrise la punctul a). **(6p.)**

77

4. Fişierul text **bac.in** conține pe prima sa linie un număr natural `n` (`0<n<10000`), iar pe următoarea linie `n` numere naturale din intervalul `[1,100]`. Se cere să se citească din fişier toate numerele şi să se afişeze pe ecran, în ordine descrescătoare, toate numerele care apar pe a doua linie a fişierului şi numărul de apariții ale fiecăruia. Dacă un număr apare de mai multe ori, el va fi afişat o singură dată. Fiecare pereche „valoare - număr de apariții" va fi afişată pe câte o linie a ecranului, numerele fiind separate printr-un spațiu, ca în exemplu. Alegeți un algoritm de rezolvare eficient din punctul de vedere al timpului de executare.

Exemplu: dacă fişierul bac.in are următorul conținut:

```
12
1 2 2 3 2 2 3 3 2 3 2 1
```

pe ecran se vor afişa, în această ordine, perechile:

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

a) Explicați în limbaj natural metoda utilizată justificând eficiența acesteia (4-6 rânduri) **(4p.)**

b) Scrieți programul `c/c++` ce rezolvă problema enunțată, corespunzător metodei descrise la punctul a). **(6p.)**

78

4. Se citeşte de pe prima linie a fişierului **numere.in** un număr natural **n** (0<n<10000) şi, de pe a doua linie a fişierului, **n** numere naturale din intervalul [1,100] şi se cere să se afişeze pe ecran, despărțite prin câte un spațiu, numărul sau numerele întregi din intervalul [1,100] care **nu** apar printre numerele citite. Dacă pe a doua linie a fişierului apar toate numerele din intervalul precizat, se va afişa mesajul **NU LIPSESTE NICIUN NUMAR**. Alegeți un algoritm de rezolvare eficient din punctul de vedere al timpului de executare.

Exemplu: pentru fişierul **numere.in** cu următorul conținut

```
12
4 2 3 1 6 5 7 8 9 11 10 100
```

se vor afişa valorile 12 13 ... 99.

a) Explicați în limbaj natural metoda utilizată, justificând eficiența acesteia (4-6 rânduri). **(4p.)**

b) Scrieți programul C/C++ ce rezolvă problema enunțată, corespunzător metodei descrise la punctul **a)**. **(6p.)**

79

4. Fişierul text **NUMERE.IN** conține, pe mai multe linii, cel mult 30000 de numere naturale nenule mai mici sau egale cu 500, despărțite prin câte un spațiu.

a) Scrieți programul C/C++ care, utilizând un algoritm eficient din punct de vedere al timpului de executare, afişează pe ecran, în ordine crescătoare, toate numerele care au apărut o singură dată din fişierul **NUMERE.IN**, despărțite prin câte un spațiu.

Exemplu: dacă fişierul **NUMERE.IN** conține numerele scrise alăturat, se vor afişa valorile următoare: 3 4 5 6 34 **(6p.)**

```
2 23 34 3
8 9 9 23
6 8 9 2
4 5 23 9
```

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

81

4. Fişierul text **NUMERE.IN** conține, pe mai multe linii, cel mult 30000 de numere naturale nenule mai mici sau egale cu 500, numerele de pe fiecare linie fiind despărțite prin câte un spațiu. Fişierul conține cel puțin două numere distincte, fiecare având două cifre.

a) Scrieți programul C/C++ care citeşte toate numerele din fişierul **NUMERE.IN** şi creează fişierul text **NUMERE.OUT** care să conțină pe prima linie cel mai mare număr de două cifre din fişierul **NUMERE.IN**, şi de câte ori apare el în acest fişier, iar pe a doua linie, cel mai mic număr de două cifre din fişierul **NUMERE.IN** şi de câte ori apare el în acest fişier. Alegeți o metodă de rezolvare eficientă din punct de vedere al memoriei utilizate şi al timpului de executare. **(6p.)**

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

Exemplu: dacă fişierul **NUMERE.IN** are conținutul alăturat:

```
2 253 34 3
6 88 9 2 3
4 54 34 88
```

atunci fişierul **NUMERE.OUT** va avea următorul conținut:

```
88 2
34 2
```

82

4. Fişierul text **NUMERE.IN** conține mai multe linii, pe fiecare linie existând câte un şir de numere naturale nenule mai mici sau egale decât 30000, despărțite prin câte un spațiu; fiecare linie se termină cu numarul 0 (care se consideră că nu face parte din şirul aflat pe linia respectivă) şi conține cel puțin două valori.

a) Scrieți programul C/C++ care afişează pe ecran valoarea maximă din şirul care conține cele mai puține numere. În cazul în care există mai multe şiruri cu acelaşi număr minim de numere, se va afişa cea mai mare valoare care apare în unul dintre aceste şiruri. Alegeți o metodă de rezolvare eficientă din punct de vedere al memoriei utilizate şi al timpului de executare. **(6p.)**

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

Exemplu: dacă fişierul **NUMERE.IN** are conținutul alăturat, atunci pe ecran se va afişa numărul 253.

```
2 253 34 3 0
6 88 9 3 0
4 54 88 12345 98 234 546 0
```

83

4. Fişierul text **bac.txt** conține cel puțin două şi cel mult 1000 de numere naturale distincte, dintre care cel puțin două sunt pare. Numerele sunt separate prin câte un spațiu şi fiecare dintre ele are cel mult 9 cifre.

a) Scrieți un program C/C++ care determină cele mai mari două numere pare din fişier, utilizând un algoritm eficient din punct de vedere al timpului de executare şi al spațiului de memorie utilizat. Cele două numere vor fi afişate pe ecran, în ordine descrescătoare, separate printr-un spațiu.
Exemplu: dacă fişierul conține numerele: 5123 8 6 12 3 se va afişa: 12 8 **(6p.)**

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

86

4. Fişierul text **bac.txt** conține pe mai multe rânduri cel mult 50000 de numere naturale din intervalul închis [0, 99], numerele de pe acelaşi rând fiind separate prin câte un spațiu.

a) Scrieți un program C/C++ care afişează pe ecran, **în ordine descrescătoare**, acele numere din fişier care sunt mai mari decât un număr natural **k**, citit de la tastatură, utilizând un algoritm eficient din punct de vedere al timpului de executare. Dacă un număr apare de mai multe ori, şi este mai mare decât **k**, se va afişa o singură dată. Numerele vor fi afişate câte 20 pe fiecare linie (cu excepția ultimei linii care poate să conțină mai puține valori), separate prin câte un spațiu.
Exemplu: dacă fişierul conține numerele: 15 8 99 15 1 37 1 24 2, iar pentru **k** se citeşte valoarea 7, se vor afişa numerele 99 37 24 15 8. **(6p.)**

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

89

4. Fişierul text **bac.txt** conține cel mult **1000** de numere întregi de cel mult **9** cifre fiecare, numerele fiind separate prin câte un spațiu; printre numerele din fişier există cel puțin două numere pozitive, aflate pe poziții consecutive.

a) Scrieți un program **c/c++** care afişează două numere pozitive, aflate unul după altul în fişier, a căror sumă este maximă, utilizând un algoritm eficient din punct de vedere al timpului de executare şi al spațiului de memorie utilizat. Dacă există mai multe soluții, se afişează doar acea pereche pentru care diferența dintre cele două numere este maximă. Numerele vor fi afişate pe ecran, în ordinea din fişier, separate printr-un spațiu.
Exemplu: dacă fişierul conține numerele: **-2 2 16 4 -1 25 -2 8 12 7 13** se vor afişa numerele **16 4**, în această ordine, cu un spațiu între ele. **(6p.)**

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

90

4. Fişierul text **număr.txt** conține pe prima linie o valoare naturală **n** cu exact 9 cifre nenule distincte. Scrieți un program eficient din punctul de vedere al timpului de executare care citeşte din fişier numărul **n** şi afişează pe ecran cea mai mică valoare **m** formată din exact aceleaşi cifre ca şi **n**, astfel încât **m>n**. În cazul în care nu există o astfel de valoare, programul va afişa pe ecran mesajul **Nu exista**.

Exemplu: Dacă fişierul **număr.txt** conține numărul **257869431**, se va afişa pe ecran numărul **257891346**.

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

b) Scrieți un program **c/c++** care rezolvă problema conform metodei descrise. **(6p.)**

94

4. **a)** Scrieți un program **c/c++** care citeşte de la tastatură un număr natural nenul, **s**, având maximum **9** cifre, şi printr-o metodă eficientă din punct de vedere al timpului de executare, determină şi scrie în fişierul **rez.dat** trei valori naturale a căror sumă este egală cu **s**, şi al căror produs este maxim. Cele trei valori vor fi scrise în ordine crescătoare pe prima linie a fişierului **rez.dat**, separate prin câte un spațiu.
Exemplu: dacă se citeşte valoarea **5**, fişierul **rez.dat** va avea o linie cu conținutul **1 2 2**. **(6p.)**

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

95

4. Fişierul **BAC.DAT** conține pe prima linie, separate printr-un spațiu, două valori naturale **n** şi **m** (**2≤n≤1000**, **2≤m≤1000**), pe a doua linie **n** valori întregi şi pe a treia linie **m** valori întregi. Valorile de pe a doua şi de pe a treia linie apar în fişier în ordine strict crescătoare, sunt separate prin câte un spațiu şi au cel mult **4** cifre fiecare.
Se cere afişarea pe ecran a două valori, dintre cele aflate în poziții consecutive pe a treia linie a fişierului, care determină intervalul închis în care se află un număr maxim de valori de pe a doua linie a fişierului. Se va utiliza o metodă eficientă din punct de vedere al timpului de executare şi al spațiului de memorie utilizat. Se garantează că cel puțin un număr aflat pe a doua linie a fişierului aparține unuia dintre intervalele determinate de numerele de pe a treia linie a fişierului.

Exemplu: dacă fişierul **BAC.DAT** are conținutul alăturat, programul va afişa: **1 9**

```
10 4
-1 1 3 4 5 6 10 15 16 117
0 1 9 20
```

Explicație: cele patru numere de pe a treia linie a fişierului determină trei intervale: **[0,1]**, **[1,9]**, **[9,20]**; în intervalul **[1,9]** se află **5** valori de pe a doua linie a fişierului, acesta fiind numărul maxim de valori aflate în unul dintre cele trei intervale.

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

b) Scrieți un program **C/C++** care să rezolve problema conform metodei descrise. **(6p.)** 96

4. Pe prima linie a fişierului text **DATE.TXT** se găseşte o valoare naturală **k** (**k≤1000000**).

a) Scrieți un program **C/C++** care citeşte din fişierul **DATE.TXT** valoarea **k** şi afişează, pe ecran, toate perechile de numere naturale nenule **x**, **y** (**x≤y**) cu proprietatea că $x^2+y^2=k$. Fiecare pereche va fi afişată pe câte o linie, numerele fiind despărțite printr-un spațiu. Alegeți o metodă de rezolvare eficientă din punctul de vedere al timpului de executare.

Exemplu: dacă fişierul **DATE.TXT** conține numărul **1000000**, pe ecran se vor afişa, nu neapărat în această ordine, perechile alăturate. **(6p.)**

```
280 960
352 936
600 800
```

b) Descrieți succint, în limbaj natural, metoda utilizată, justificând eficiența acesteia **(4p.)** 97

4. Pe prima linie a fişierului text **DATE.TXT** se află două numere naturale nenule **n** şi **m** (**n≤3000**, **m≤3000**), pe a doua linie un şir de **n** numere naturale, ordonate crescător, având fiecare cel mult **9** cifre, iar pe linia a treia un şir de **m** numere naturale, ordonate descrescător, având fiecare cel mult **9** cifre. Numerele sunt despărțite, în cadrul liniilor, prin câte un spațiu.

a) Scrieți programul **C/C++** care citeşte numerele din fişier şi afişează, pe ecran, doar numerele pare din cele două şiruri, ordonate crescător. Programul nu va afişa nimic dacă nu există numere pare în cele două şiruri. Alegeți o metodă de rezolvare eficientă ca timp de executare.

Exemplu: dacă fişierul are conținutul alăturat, pe ecran se va afişa: **2 4 4 32 42 42 88 88** **(6p.)**

```
5 8
2 4 7 37 42
88 88 67 45 42 32 4 1
```

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

98

4. Pe prima linie a fişierului text **DATE.TXT** se află un şir de cel mult **10000** de numere întregi, având cel mult **4** cifre fiecare. Numerele sunt despărțite prin câte un spațiu.

a) Scrieți un program **C/C++** care citeşte numerele din fişier şi afişează pe ecran lungimea maximă a unei secvențe de numere din şir, cu proprietatea că oricare două numere din secvență, aflate pe poziții consecutive, au parități diferite. Pe a doua linie a ecranului, programul va afişa o secvență de lungime maximă, valorile fiind despărțite prin câte un spațiu. Dacă există mai multe secvențe de lungime maximă, se va afişa una dintre ele, oricare. Alegeți o metodă de rezolvare eficientă ca timp de executare.

Exemplu: dacă fişierul conține, în ordine, numerele **2 4 3 2 7 4 6 2 7 8 12**, se va afişa:

5

4 3 2 7 4 **(6p.)**

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

99

4. Se consideră un şir **s** format după regula alăturată, unde s-a notat cu **aΘb** numărul obținut prin concatenarea cifrelor lui **a** şi **b**, în această ordine.

$$s_n=\begin{cases} x & \text{dacă } n=1 \\ x+1 & \text{dacă } n=2 \\ s_{n-1}\Theta s_{n-2} & \text{dacă } n>2 \end{cases}$$

Exemplu: pentru **x**=2 se obține şirul:

2, 3, 32, 323, 32332,....

Fişierul text **SIR.TXT** conține pe prima linie două numere, **x** (**1≤x≤20**) şi **k** (**1≤k≤5000**), separate printr-un spațiu, iar pe a doua linie un număr format din exact **k** cifre, reprezentând un termen al şirului **s** (diferit de **x**). Cifrele numărului **nu** sunt separate prin spații.

a) Scrieți un program **C/C++** care, utilizând un algoritm eficient din punct de vedere al timpului de executare şi al memoriei utilizate, afişează pe ecran acel termen din şir care îl precede pe cel citit din fişier.

Exemplu: dacă fişierul conține valorile alăturate, se va afişa pe ecran numărul **323**. **(6p.)**

```
2 5
32332
```

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

100