FIŞA DE LABORATOR 3
Vectori caracteristici şi vectori de frecvenţă

1.
Fişierul text **numere.txt** conţine pe prima linie un număr natural **n** (**0<n<100000**), iar pe a doua linie, separate prin câte un spaţiu, **n** numere naturale formate din cel mult două cifre.

Scrieţi un program **C/C++** care determină în mod eficient, din punct de vedere al timpului de executare, toate numerele conţinute de a doua linie a fişierului care apar de cel puţin două ori în acestă linie. Programul va afişa pe ecran numerele determinate, o singură dată, în ordine crescatoare, pe aceeaşi linie, separate prin câte un spaţiu.

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

```
44 2 54 74 2 44 9 2
```

atunci pe ecran se va afişa: **2 44**

2.
Se citeşte de pe prima linie a fişierului text **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, despărţite prin câte un spaţiu, 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**. Alegeţi un algoritm de rezolvare eficient din punctul de vedere al timpului de executare. Scrieţi programul **C/C++** ce rezolvă problema enunţată, corespunzător metodei descrise.

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

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

se vor afişa valorile **12 13 ... 99**.

3.
Fişierul text **bac.txt** conţine 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 spaţiu.

Scrieţi un program **C/C++** care afişează pe ecran, **în ordine crescătoare**, 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 care corespunde cerinţei apare de mai multe ori, se va afişa o singură dată. Numerele vor fi afişate pe ecran separate prin câte un spaţiu.

Exemplu: dacă fişierul conţine numerele: **15 36 33 36 1 12 1 24 2**, iar pentru **k** se citeşte valoarea **24**, se vor afişa numerele **33 36**.

4.
Pe prima linie a fişierului text **DATE.TXT** se află un număr natural **n (0<n≤10000)**, iar pe a doua linie un şir de **n** numere naturale, depărţite prin câte un spaţiu, fiecare având cel mult **4** cifre.

Scrieţi un program **C/C++** care citeşte numerele din fişier şi afişează, pe ecran, valorile din şir, în ordinea crescătoare a cifrei unităţilor. Dacă două numere din şir au aceeaşi cifră a unităţilor nu contează care dintre ele va fi afişat primul. Realizaţi un program eficient din punct de vedere al timpului de executare.

Exemplu: dacă fişierul **DATE.TXT** conţine

```
7
32 491 26 328 213 500 422
```

pe ecran se va afişa:

```
500 491 32 422 213 26 328
```

5.

Pe prima linie a fişierului text **DATE.TXT** se află un şir de cel mult **10000** de numere naturale, despărțite prin câte un spațiu, fiecare având **exact o cifră**.

Scrieți un program **C/C++** care citeşte numerele din fişier şi le scrie în fişierul text **OUT.TXT**, pe o singură linie, în ordine crescătoare a valorilor lor, separate prin câte un spațiu. Se va utiliza un algoritm eficient din punct de vedere al timpului de executare.

Exemplu: dacă din fişierul **DATE.TXT** se citeşte şirul:

```
2 4 3 2 7 4 3 7 2 7 7 2 1 9 1 1 2 3
```

fişierul **OUT.TXT** va conține

```
1 1 1 2 2 2 2 2 3 3 3 4 4 7 7 7 7 9
```

6.

Fișierul **bac.txt** conține numere naturale din intervalul $[1,10^4]$: pe prima linie numărul **n**, pe a doua linie un șir de **n** numere ordonate strict descrescător, iar pe a treia linie două numere, **x** și **y** ($x \leq y$). Numerele de pe aceeași linie sunt separate prin câte un spațiu.

Se cere să se afișeze pe ecran cel mai mare număr din șir care aparține intervalului **[x,y]**. Dacă nu există un astfel de număr, se afișează pe ecran mesajul **nu exista**. Pentru determinarea numărului cerut se utilizează un algoritm eficient din punctul de vedere al timpului de executare.

Scrieți programul **C/C++** corespunzător algoritmului descris.

Exemplu: dacă fișierul conține numerele

```
5
100 49 16 7 2
10 30
```

atunci pe ecran se afișează

```
16
```

7.

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 două 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.

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
```

8.

Un număr **x** se numește **sufix** al unui număr **y** dacă **y** se poate obține din **x** prin alipirea, la stânga sa, a cel puțin unei alte cifre.

Fişierul **bac.in** conține un şir de cel mult 10^9 numere naturale din intervalul $[0,10^9]$. Numerele din şir sunt separate prin câte un spaţiu.

Se cere să se afișeze pe ecran, în ordine strict crescătoare, toate numerele din intervalul $[100,999]$ care sunt termeni ai șirului aflat în fișier și sunt sufixe pentru cel puțin un alt termen al aceluiași șir. Numerele sunt afișate câte unul pe linie, iar dacă în șir nu există astfel de numere, se afișează pe ecran mesajul **Nu exista**. Pentru determinarea numerelor cerute se utilizează un algoritm eficient din punctul de vedere al timpului de executare.

Scrieţi programul C/C++ corespunzător algoritmului descris.

Exemplu: dacă fişierul **bac.in** are conţinutul

```
15502 49 54321 6149 76149 123 123 502 4321 321 321
```

atunci pe ecran se afişează numerele

```
321
502
```

9.

Un șir de numere este o **progresie aritmetică de rație r** dacă oricare termen al său, cu excepția primului, se obține din cel care îl precede, prin adunarea la acesta a numărului **r**.

Exemplu: șirul 12, 14, 16, 18, 20 este o progresie de rație 2.

Fişierul **bac.in** conține un șir de cel mult 10^6 numere naturale din intervalul $[0,10^3]$, separate prin câte un spațiu.

Se cere să se verifice dacă există un număr natural **r**, astfel încât toate numerele **distincte** din șir să poată fi rearanjate, pentru a forma o progresie aritmetică de rație **r**. Se afișează pe ecran numărul **r**, sau mesajul **NU**, dacă nu există un astfel de număr. Proiectați un algoritm eficient din punctul de vedere al timpului de executare.

Scrieţi programul C/C++ corespunzător algoritmului descris.

Exemplu: dacă fișierul conține numerele

```
180 30 80 280 130 330 230 30 30 330 80
```

se afișează pe ecran

```
50
```

10.

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

Scrieți un program **C/C++** care afişează pe ecran, în mod eficient din punct de vedere al timpului de executare, toate numerele situate pe a doua linie a fişierului, în ordinea crescătoare a valorilor lor, separate prin câte un spațiu. Dacă un număr apare în fişier de mai multe ori el va fi afişat o singură dată.

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

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