Şiruri prelucrate dintr-o singură traversare (direct la citire)

1. Fişierul **bac.txt** conţine: pe prima linie numărul n (natural, par) iar pe linia a doua un şir cu n numere naturale. Numerele din şir sunt în ordine crescătoare şi sunt scrise în fişier separate prin spaţiu. Să se afişeze pe ecran cel mai mare număr din a doua jumătate a şirului care este mai mic decât oricare dintre numerele aflate în a doua jumătate. Dacă în fişier nu se află o asemenea valoare, pe ecran se afişează mesajul NU EXISTA.

   Pentru determinarea numărului cerut se va folosi un algoritm eficient ca timp de executare şi ca spaţiu de memorare.

   Exemple:

dacă fişierul **bac.txt** are conţinutul alăturat (numărul 5 apare de 26 de ori) programul va afişa valoarea 3	**30** **1 3 5 5 5 .... 5 7 10**
dacă fişierul **bac.txt** are conţinutul alăturat programul va afişa NU EXISTA	**6** **3 3 3 3 9 15**

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    int n, i;
    int p,u;     // ultimele doua numere distincte, din prima jumatate
    int x;       // numar citit
    ifstream f("bac1.txt");
    f>>n;
    n=n/2;       // citesc doar prima jumatate a sirului
    p=u=-1;
    for(i=1; i<=n;i++) {
        f>>x;
        if (x>u) {
            p=u;
            u=x;
        }
    }
    // citesc doar primul numar din a doua jumatate
    f>>x;
    f.close();
    // afisare
    if (x>u) cout<<u;
    else
        if(x>p && p>-1) cout<<p;
        else cout<<"NU EXISTA";
    return 0;
}
```

2. Numim **secvență uniformă** a unui șir de numere naturale, un subșir al acestuia, format din termeni cu aceeași valoare, aflați pe poziții consecutive în șir. Lungimea secvenței este egală cu numărul de termeni ai acesteia.
   Fișierul **bac.txt** conține un șir cu cel puțin două și cel mult 10^8 numere naturale din intervalul $[0,10^9]$. Numerele sunt scrise separate prin spațiu iar în șir există cel puțin doi termeni egali, pe poziții consecutive. Să se determine o secvență uniformă de lungime maximă, în șirul aflat în fișier și să se afișeze pe ecran: lungimea acesteia și pe o linie nouă, separați prin spațiu, termenii acesteia. Dacă șirul conține mai multe astfel de secvențe, se vor afișa doar termenii ultimei dintre acestea.

Pentru determinarea numerelor cerute se va folosi un algoritm eficient ca timp de executare și ca spațiu de memorare.

Exemplu:

dacă fișierul **bac.txt** are conținutul alăturat pe ecran se afișează liniile de mai jos: **4** **11 11 11 11**	**2 3 3 3 3 4 4 17 17 17 17 16 11 11 11 11 15 15**

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    int Max=0;       // lungimea secventei uniforme de lungime maxima
    int a;           // numarul din care este alcatuita secventa
    int l;           // lungimea secventei uniforme in care ne gasim
    int p,u;         // penultimul si ultimul numar citit
    ifstream f("bac.txt");
    p=-1;
    while (f>>u) {
        if (u==p) l++;
        else l=1;
        p=u;
        if (l>=Max) {
            Max=l;  a=u;
        }
    }
    f.close();
    // afisare
    cout<<Max<<endl;
    for( int i=1; i<=Max; i++)
        cout<<a<<' ';
    return 0;
}
```

3. Fişierul text **date.in** conţine un şir cu cel mult un milion de numere naturale din intervalul $[0,10^3]$, separate prin spaţiu. Şirul are cel puţin doi termeni pari şi cel puţin doi termeni impari. Să se afişeze pe ecran mesajul DA dacă şirul aflat în fişier are un subşir ordonat strict crescător, format din toţi termenii impari ai săi şi un subşir ordonat strict descrescător format din toţi termenii pari ai săi. Dacă nu există două astfel de subşiruri, programul afişează pe ecran mesajul NU.

   Pentru verificarea proprietăţii cerute se va folosi un algoritm eficient ca timp de executare şi ca spaţiu de memorare.

   Exemple:

dacă fişierul **date.in** are conţinutul alăturat pe ecran se afişează mesajul DA	**8 1 6 3 5 4 7**
dacă fişierul **date.in** are conţinutul alăturat pe ecran se afişează mesajul NU	**2 1 6 3 5 4 7**

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    int up=1001;     // ultimul numar par din fisier
    int ui=-1;       // ultimul numar impar din fisier
    int x;           // numar citit
    int ok=1;        // presupun ca exista cele doua subsiruri
    ifstream f("date.in");
    while (f>>x)
        if (x%2==0)
            if (x<up) up=x;
            else {
                ok=0; break;
            }
        else
            if (x>ui) ui=x;
            else {
                ok=0; break;
            }
    f.close();
    // afisare
    if (ok==1) cout<<"DA";
    else cout<<"NU";
    return 0;
}
```

4. Fişierul text **numere.in** conţine: pe prima linie numărul natural n, $n \in [2, 10^9]$ iar pe linia a doua un şir cu cel mult 10^8 numere naturale din intervalul [1,n]. Numerele din şir sunt ordonate crescător şi sunt separate prin câte un spaţiu. Să se determine valorile naturale distincte din intervalul [1,n] care NU se regăsesc în şirul din menţionat mai sus. Valorile determinate se vor afişa pe ecran, în ordine crescătoare, separate prin spaţiu. Dacă toate valorile naturale din intervalul [1,n] sunt scrise în fişier, pe ecran mesajul NU EXISTA.

   Pentru determinarea valorilor cerute se va folosi un algoritm eficient ca timp de executare şi ca spaţiu de memorare.

   Exemplu:

dacă fişierul **numere.in** are conţinutul alăturat pe ecran se afişează linia de mai jos **1 2 5 6 7 9 10**	**10** **3 4 4 8**

```cpp
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    int p;      // penultimul numar citit din fisier
    int u;      // ultimul numar citit din fisier
    int nr=0;   // numarul de valori afisate
    int n,i;
    ifstream f("numere.in");
    f>>n;
    f>>p;
    // afisez numerele din [1,n], mai mici decat primul citit
    for(i=1; i<p;i++) {
        cout<<i<<' ';
        nr++;
    }
    // parcurg restul fisierului
    while (f>>u) {
        // afisez toate numerele din [p+1, u-1]
        for(i=p+1; i<=u-1;i++) {
            cout<<i<<' ';
            nr++;
        }
        p=u;
    }
    // afisez numerele din [1,n], mai mari decat ultimul citit
    for(i=u+1; i<=n;i++) {
        cout<<i<<' ';
        nr++;
    }
    f.close();
    // final
    if (nr==0) cout<<"NU EXISTA";
    return 0;
}
```

5. Fişierul **IN.txt** conţine cel puţin două şi cel mult 10000 de numere naturale, scrise în ordine crescătoare şi separate prin spaţiu. Cel puţin un număr din fişier apare o singură dată. Să se citească întreg conţinutul fişierului şi să se afişeze pe ecran , în ordine crescătoare şi separate prin spaţiu, numai numerele care apar în fişier o singură dată.

   Pentru determinarea valorilor cerute se va folosi un algoritm eficient ca timp de executare şi ca spaţiu de memorare.

   Exemplu:

dacă fişierul **IN.txt** are conţinutul alăturat pe ecran se afişează linia de mai jos **7 21**	**1 1 2 2 2 7 10 10 10 21**

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    int a,b ;    // ultimele doua numere diferite, citite din fisier
    int nr;      // numarul de aparitii in fisier ale numarului a
    ifstream f("IN.txt");
    f>>a; nr=1;
    while (f>>b)
        if (b==a) nr++;
        else {
            if (nr==1) cout<<a<<' ';
            nr=1;
            a=b;
        }
    f.close();
    if (nr==1) cout<<b;
    return 0;
}
```

6. Fiind date două numere naturale **a** şi **b**, îl numim pe **a** sufix al lui **b** dacă este egal cu **b** sau dacă **b** se poate obţine din **a** prin alipirea la stânga a unor cifre. Exemple: 12 este sufix al lui 12 iar 15 este sufix al lui 31415.
   Fişierul **bac.txt** conţine: pe prima linie un număr natural **x** şi pe linia a doua un şir cu cel puţin două şi cel mult 10^8 numere naturale. Numerele din şir sunt separate prin câte un spaţiu. Să se afişeze pe ecran ultimul termen al şirului care este sufix al numărului **x**. Dacă în şir nu apare o asemenea valoare pe ecran se afişează mesajul NU EXISTA.

   Exemplu:

dacă fişierul **bac.txt** are conţinutul alăturat pe ecran se afişează numărul **312**	**12** **32112 1245 12 67120 312 12345**

```cpp
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    int x;
    int y;       // numar citit din fisier, de pe linia 2
    int u=-1;    // numarul cerut
    ifstream f("bac.txt");
    f>>x;
    // calculez in p, 10 la puterea k, unde k=numarul de cifre din x
    int a=x, p=1;
    while (a>0)  {
        p=p*10;
        a=a/10;
    }
    // citesc sirul de pe linia 2 a fisierului
    while (f>>y)
        if (y%p==x) u=y;
    f.close();
    // afisare
    if (u==-1) cout<<"NU EXISTA";
    else cout<<u;
    return 0;
}
```

7. În fişierul **IN.txt** se află, pe mai multe linii, mai multe numere întregi. Pe fiecare linie există cel puţin un număr şi oricare două numere sunt separate prin spaţiu. Să se afişeze, pe câte o linie, în fişierul **OUT.txt**, prima şi ultima valoare de pe fiecare linie din fişierul **IN.txt**, separate prin spaţiu.

Exemplu:

dacă fişierul **IN.TXT** are conţinutul alăturat atunci fişierul **OUT.txt** va conţine liniile de mai jos	**12 34 53 51** **23 567 89 1 34** **12** **45 792 3**
12 51 **23 34** **12 12** **45 3**	

```cpp
#include <fstream>
using namespace std;
int main(){
    int x;       // numar citit din fisier
    int nr=0;    // numarul de valori citite pe pe linia pe care ma gasesc
    ifstream f("IN.txt");
    ofstream g("OUT.txt");
    while (f.peek()!=EOF) {
        f>>x;
        nr++;
        if (nr==1) g<<x<<' ';
        if (f.peek()=='\n') {
            g<<x<<endl;
            nr=0;
        }
    }
    g<<x<<endl; // la ultima linie, nu mai intalneste ENTER
    g.close();
    f.close();
    return 0;
}
```

8. În fişierul **IN.txt** se află, pe mai multe linii, mai multe numere întregi. Pe fiecare linie există cel puţin un număr şi oricare două numere sunt separate prin spaţiu. Să se afişeze, în fişierul **OUT.txt**, maximul de pe fiecare linie a fişierului **IN.txt**. Fişierul **OUT.txt** va conţine numere separate prin spaţiu.

Exemplu:

dacă fişierul **IN.TXT** are conţinutul alăturat atunci fişierul **OUT.txt** va conţine linia de mai jos **53 567 -12 45**	**12 34 53 51** **23 567 89 1 34** **-12** **45 -792 3**

```
#include <fstream>
using namespace std;
int main(){
    int Max;     // maximul pe fiecare linie
    int x;       // numar citit din fisier
    int nr=0;    // numarul de valori citite de pe linia pe care ma aflu
    ifstream f("IN.txt");
    ofstream g("OUT.txt");
    while (f.peek()!=EOF) {
        f>>x;
        nr++;
        if (nr==1) Max=x;
        else
            if (x>Max) Max=x;
        if (f.peek()=='\n') {
            g<<Max<<' ';
            nr=0;
        }
    }
    g<<Max;      // la ultima linie nu se mai intalneste ENTER
    g.close();
    f.close();
    return 0;
}
```

Afișarea numerelor prin generarea cifrelor

9. Se citesc, de la tastatură, numerele naturale **s1** și **s2** (0<s1≤18, 0<s2≤18). Să se afișeze, în ordine crescătoare, toate numerele naturale cu exact 5 cifre, pentru care suma primelor două cifre este chiar **s1** și suma ultimelor două cifre este chiar **s2**. Numerele se vor afișa, fiecare pe câte o linie, în fișierul **bac.txt**.

Exemplu:

dacă s1=8 și s2=7 atunci unul dintre numerele ce are proprietatea cerută este 35725 deoarece 3+5=8 și 2+5=7.

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
      int s1, s2;
      int a,b,c,d,e;   // cifrele unui numar
      cout << "s1,s2" << endl;
      cin>>s1>>s2;
      ofstream g("bac.txt");
      for(a=1;a<=9;a++) {
            b=s1-a;
            if (b>=0 && b<=9)
                  for(c=0;c<=9;c++)
                        for(d=0;d<=9;d++) {
                              e=s2-d;
                              if (e>=0 && e<=9)
                                    g<<a<<b<<c<<d<<e<<endl;
                        }
      }
      g.close();
      return 0;
}
```

Varianta 2

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
      int s1, s2;
      cout << "s1,s2" << endl;
      cin>>s1>>s2;
      ofstream g("bac.txt");
      int x;
      for(x=10000;x<=99999;x++)
            if (x/10000+x/1000%10==s1 && x/10%10+x%10==s2)
                  g<<x<<endl;
      g.close();
      return 0;
}
```