#877 Cuburi2

CERINȚA

Gigel are un set de n cuburi. Fiecare cub este marcat cu un număr natural, de la 1 la n și i se cunoaște lungimea laturii - număr natural. Cu o parte dintre aceste cuburi Gigel va construi o stivă, astfel:

- fiecare cub se analizează o singură dată, în ordinea numerelor marcate;
- dacă stiva nu conține niciun cub, cubul curent devine baza stivei
- dacă cubul curent are latura mai mică sau egală cu cubul din vârful stivei, se adaugă pe stivă;
- dacă cubul curent are latura mai mare decât cubul din vârful stivei, se vor înlătura de pe stivă cuburi (eventual toate) până când cubul curent are latura mai mică sau egală cu cubul din vârful stivei.

Să se afișeze numerele de pe cuburile existente la final în stivă, de la bază spre vârf.

DATE DE INTRARE

Programul citește de la tastatură numărul n, apoi n numere naturale, reprezentând, în ordine, lungimile laturilor cuburilor.

DATE DE IEȘIRE

Programul va afișa pe ecran numărul de cuburi existente pe stivă, iar pe linia următoare, separate prin câte un spațiu, numerele marcate pe aceste cuburi.

RESTRICȚII ȘI PRECIZĂRI

$1 \leq n \leq 1000$

lungimile cuburilor vor fi mai mici decât 1000

Exemplu:

Intrare

6

7 4 3 5 1 2

Ieșire

3

1 4 6

Explicație

Operațiile efectuate sunt:

- se adaugă cubul 1. Stiva devine 1
- se adaugă cubul 2. Stiva devine 1 2
- se adaugă cubul 3. Stiva devine 1 2 3
- pentru a adaugă cubul 4, trebuie eliminate cuburile 2 3. Stiva devine 1 4
- se adaugă cubul 5. Stiva devine 1 4 5
- pentru a adaugă cubul 6, trebuie eliminat cubul 5. Stiva devine 1 4 6

REZOLVARE

```
#include <iostream>
using namespace std;
int stiva[1001][2],i,n,x,k;
int main()
{
    cin>>n;
    for(i=1; i<=n; i++)
    {
        cin>>x;
        while(x>stiva[k][0]&&k>0)
            k--;
        k++;
        stiva[k][0]=x;
        stiva[k][1]=i;
    }
    cout<<k<<'\n';
    for(i=1; i<=k; i++)
        cout<<stiva[i][1]<<" ";
    return 0;
}
```

#2650 books

Eroul nostru, Căldăruşe, are un număr n de cărţi pe care le are aranjate una peste cealaltă (sub forma unui stack). Cartea din vârf are valoarea a1, următoarea a2 şi aşa mai departe. Cartea de la bază are indicele n (an). Toate numerele sunt distincte.

Căldăruşe vrea să mute toate cărţile în ghiozdanul lui în exact n paşi. În timpul pasului de ordin i, el vrea să mute cartea cu numărul bi în ghiozdan. Dacă această carte se află în stack, el o ia atât pe ea, cât şi toate cărţile situate deasupra acesteia şi le pune în ghiozdan; în caz contrar, el nu va face nimic şi va trece la următorul pas. De exemplu, dacă în stack cărţile sunt aranjate în ordinea [1, 2, 3] (cartea cu numărul 1 este aflată în vârf) şi paşii prin care Căldăruşe trece sunt, în această ordine, [2, 1, 3], atunci în cadrul primului pas el va muta două cărţi (1 şi 2), în cadrul celui de-al doilea pas nu va face nimic (din moment ce cartea cu numărul 1 este deja în ghiozdan) şi în cadrul ultimului pas va muta o singură carte (cartea cu numărul 3).

CERINŢA

Ajutaţi-l pe Căldăruşe! Spuneţi-i voi numărul de cărţi pe care le va pune în ghiozdan în timpul fiecărui pas.

DATE DE INTRARE

Prima linie va conţine numărul n, cu semnificaţia din enunţ. Următoarea linie va conţine n numere întregi, anume a1,a2,…,an. A treia linie va conţine alte n numere întregi, anume b1,b2,…,bn.

DATE DE IEŞIRE

Programul va afişa pe ecran n numere întregi. Elementul de ordine i ar trebui să fie egal cu numărul de cărţi pe care Căldăruşe le mută în ghiozdanul său în timpul pasului i.

RESTRICŢII ŞI PRECIZĂRI

- $1 \le a1,a2,\ldots,an,b1,b2,\ldots,bn \le n \le 200.000$;
- Numerele a1,a2,…,an sunt distincte între ele;
- Numerele b1,b2,…,bn sunt distincte între ele.

Exemplul 1:

Intrare
3
1 2 3
2 1 3
Ieşire
2 0 1

Exemplul 2:

Intrare
5
3 1 4 2 5
4 5 1 3 2
Ieşire
3 2 0 0 0

Exemplul 3:

Intrare
6
6 5 4 3 2 1
6 5 3 4 2 1
Ieșire
1 1 2 0 1 1

Explicație

Primul exemplu este descris în enunț. În cel de-al doilea exemplu, în timpul primului pas, Căldăruşe va muta cărţile [3, 1, 4]. După aceea, numai cărţile 2 şi 5 vor rămâne în stack (cartea 2 fiind deasupra cărţii 5). În timpul celui de-al doilea pas, Căldăruşe va lua şi cărţile 2 şi 5. După aceea, stackul rămâne gol, şi deci în timpul următorilor paşi, Căldăruşe nu va mai muta nicio carte.

REZOLVARE

```
#include <iostream>

using namespace std;

int main()
{
    int n, v[200001], currpos = 0, i, j, x;
    bool stiva[200001]{};

    cin >> n;
    for(i = 0; i < n; i++)
        cin >> v[i];

    for(i = 0; i < n; i++)
    {
        cin >> x; j = currpos;

        if(!stiva[x])
        {
            j = currpos;
            while(v[j] != x) { stiva[v[j]] = true; j++; }
            cout << j - currpos + 1;
            currpos = j + 1;
        }
        else cout.put('0');

        cout.put(' ');
    }
    return 0;
}
```

#878 Intervale4

CERINȚA

Se consideră un șir de n intervale închise întregi. Două intervale consecutive în șir care au intersecția nevidă se reunesc și se înlocuiesc în șir cu intervalul reuniune. Operația se repetă până când nu mai sunt în șir două intervale consecutive cu intersecția nevidă. Să se determine câte intervale există în șir după realizarea acestor operații.

DATE DE INTRARE

Fișierul de intrare intervale4.in conține pe prima linie numărul n, iar următoarele n linii câte două valori x y, reprezentând intervalele inițiale, în ordine.

DATE DE IEȘIRE

Fișierul de ieșire intervale4.out va conține pe prima linie numărul C de intervale din șir, după realizarea operațiilor.

RESTRICȚII ȘI PRECIZĂRI

- 1 ≤ n ≤ 100.000
- pentru fiecare interval dat, x ≤ y

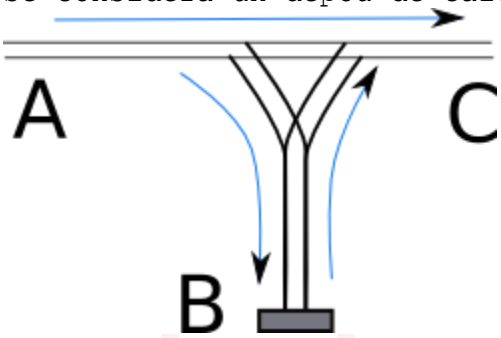
REZOLVARE

```
#include <iostream>
#include <fstream>
using namespace std;

ifstream fin ("intervale4.in");
ofstream fout("intervale4.out");

int A[100005] , B[100005] , n , nrs;

bool Intersectie(int a , int b , int x ,
int y)
{
    if(a <= x && x <= b)
        return true;
    if(a <= y && y <= b)
        return true;
    if(x <= a && a <= y)
        return true;
    if(x <= b && b <= y)
        return true;
    return false;
}
int main()
{
    fin >> n;
    nrs = 0;
    for(int i = 1 ; i <= n ; i ++)
    {
        nrs ++;
        fin >> A[nrs] >> B[nrs];
        while(nrs > 1 &&
Intersectie(A[nrs-1] , B[nrs-1] , A[nrs]
, B[nrs]))
        {
            if(A[nrs-1] > A[nrs])
                A[nrs-1] = A[nrs];
            if(B[nrs-1] < B[nrs])
                B[nrs-1] = B[nrs];
            nrs --;
        }
    }
    fout << nrs << "\n";
    return 0;
}
```

#870 Depou

CERINȚA

Se consideră un depou de cale ferată precum cel din imagine:



Pe linia A se află n vagoane, numerotate cu valori distincte de la 1 la n, într-o ordine oarecare. Vagoanele trebuie mutate pe linia C, în ordinea 1 2 .. n. Pentru aceasta se poate muta câte un vagon de pe o linie pe alta, în ordinea indicată de săgeți:

- A -> B,
- A -> C
- B -> C.

Să se determine o succesiune de operații care să mute toate vagoanele de pe linia A pe linia C în ordinea dorită.

DATE DE INTRARE

Programul citește de la tastatură numărul n, iar apoi n numere naturale distincte cuprinse între 1 și n, reprezentând ordinea vagoanelor de pe linia A.

DATE DE IEȘIRE

Programul va afișa pe ecran numărul C de operații efectuate, apoi cele C operații. Fiecare operație va fi afișată pe câte o linie a ecranului, și va consta din două caractere de forma X Y, semnificând faptul că se mută un vagon de pe linia X pe linia Y. Dacă nu este posibilă mutarea vagoanelor de pe linia A pe linia C, numărul de operații afișat va fi 0.

RESTRICȚII ȘI PRECIZĂRI

- $1 \leq n \leq 1000$

Exemplu

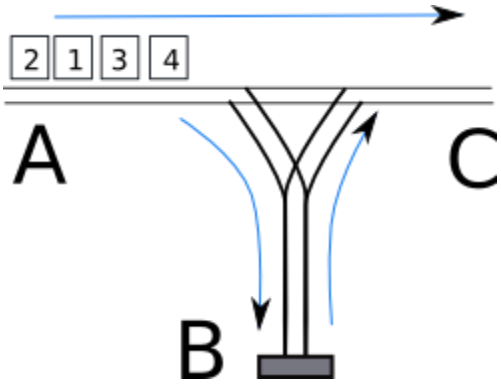
Intrare

```
4
2 1 3 4
```

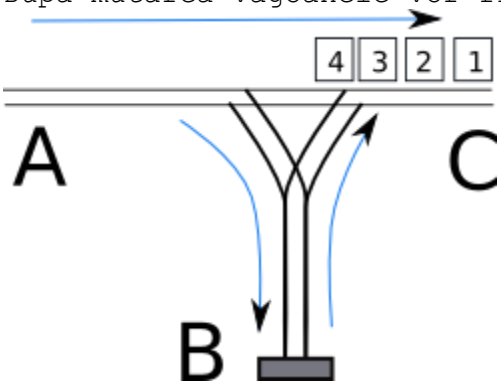
Ieșire

```
6
A B
A B
A C
A C
B C
B C
```

Explicație
Ordinea inițială a vagoanelor este următoarea:



După mutarea vagoanele vor fi în ordinea:



REZOLVARE

```
#include <iostream>
#include <cmath>
#define DIM_MAX 2000
using namespace std;

int is_FULL (int st[], int vf)
{
    return (vf==DIM_MAX);
}
int is_EMPTY (int st[], int vf)
{
    return (vf==0);
}
void PUSH (int st[], int &vf, int val)
{
    if(is_FULL(st, vf)==0)
        st[++vf]=val;
    else
        cout<<"Nu mai este spatiu pentru adaugare!\n";
}
```

```
void POP (int st[], int &vf, int &val)
{
    if(is_EMPTY(st, vf)==0)
        val=st[vf--];
    else
        cout<<"Nu putem sterge!\n";
}
int pp(int n)
{
    if(sqrt(n)==int(sqrt(n)))
        return 1;
    return 0;
}
int pc(int n)
{
    while(n>=10)
        n/=10;
    return n;
}
int ucpc(int n)
{
    if(pc(n)==n%10)
        return 1;
    return 0;
}
int ver(int a[], int n, int val)
{
    for(int i=0; i<n; i++)
    {
        if(a[i]==val)
            return 1;
    }
    return 0;
}
int st1[DIM_MAX], st2[DIM_MAX], st3[DIM_MAX], val1, val2, vf1, vf2, vf3, v1, v2,
n, val, sw=0;
int sortare(int a[], int n)
{
    for(int i=0; i<n-1; i++)
    {
        if(a[i]<a[i+1])
            return 0;
    }
    return 1;
}
char k[3000][3];
int main ()
{
    int q=0;
    cin>>n;
```

```
    for(int i=0; i<n; i++)
        cin>>val, PUSH(st1, vf1, val);
    for(int i=1; i<=n; i++)
    {
        if(ver(st1, vf1, i))
        {
            while(st1[vf1] && st1[vf1]!=i)
            {
                POP(st1, vf1, val);
                PUSH(st2, vf2, val);
                k[q][1]='A', k[q][2]='B', q++;
            }
        }
        /*if(ver(st2, vf2, i))
        {
            while(st2[vf2] && st2[vf2]!=i)
            {
                POP(st2, vf2, val);
                PUSH(st1, vf1, val);
                k[q][1]='B', k[q][2]='A', q++;
            }
        }*/
        if(st1[vf1]==i)
        {
            POP(st1, vf1, val);
            PUSH(st3, vf3, val);
            k[q][1]='A', k[q][2]='C', q++;
            sw=1;
        }
        else
        {
            if(st2[vf2]==i)
            {
                POP(st2, vf2, val);
                PUSH(st3, vf3, val);
                k[q][1]='B', k[q][2]='C', q++;
                sw=1;
            }
            else
            {
                cout<<0;
                return 0;
            }
        }
    }
    cout<<q<<"\n";
    for(int i=0; i<q; i++)
        cout<<k[i][1]<<" "<<k[i][2]<<"\n";
    return 0;
}
```

#848 Paranteze1

CERINȚA

Se dau n șiruri de paranteze rotunde. Să se stabilească, despre fiecare șir, dacă este corect parantezat – adică dacă parantezele se închid corect.
Un șir de paranteze S rotunde este corect parantezat dacă:
S este șirul vid, sau
S = (T) și T este corect parantezat, sau
S = AB, iar A și B sunt corect parantezate.
Date de intrare
Fișierul de intrare paranteze1.in conține pe prima linie numărul n, pe următoarele n linii câte un șir de paranteze rotunde.

DATE DE IEȘIRE

Fișierul de ieșire paranteze1.out va conține n linii: fiecare linie va conține valoarea 1, dacă șirul corespunzător de paranteze este corect parantezat și 0 în caz contrar.

RESTRICȚII ȘI PRECIZĂRI

- $1 \leq n \leq 100$
- fiecare șir va avea cel mult 255 de paranteze

Exemplu

paranteze1.in

```
4
(())
)(()
()((()())())
()(
```

paranteze1.out

```
1
0
1
0
```

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

ifstream fin("paranteze1.in");
ofstream fout("paranteze1.out");

int n;
char s[256];

int main()
{
    fin >> n;
    for(int k = 1 ; k <= n ; k ++)
    {
        fin >> s;
        int ok = 1;
        int nivel = 0;
        for(int i = 0 ; s[i] && ok ; i ++)
            if(s[i] == '(')
                nivel ++;
            else
                if(nivel > 0)
                    nivel --;
                else
                    ok = 0;
        if(nivel > 0)
            ok = 0;
        fout << ok << "\n";
    }
}
```

#852 Paranteze3

CERINȚA

Se dau n șiruri de paranteze rotunde sau pătrate. Să se stabilească, despre fiecare șir, dacă este corect parantezat – adică dacă parantezele se închid corect.
Un șir de paranteze S rotunde este corect parantezat dacă:
S este șirul vid, sau
S = (T) și T este corect parantezat, sau
S = [T] și T este corect parantezat, sau
S = AB, iar A și B sunt corect parantezate.
Date de intrare
Fișierul de intrare paranteze3.in conține pe prima linie numărul n, pe următoarele n linii câte un șir de paranteze rotunde sau pătrate.

DATE DE IEȘIRE

Fișierul de ieșire paranteze3.out va conține n linii: fiecare linie va conține valoarea 1, dacă șirul corespunzător de paranteze este corect parantezat și 0 în caz contrar.

RESTRICȚII ȘI PRECIZĂRI

- 1 ≤ n ≤ 100
- fiecare șir va avea cel mult 255 de paranteze

Exemplu

paranteze3.in	paranteze3.out
4	
()[]	1
)([]	0
()[(()[])()]	1
([)]	0

REZOLVARE

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

ifstream fin("paranteze3.in");
ofstream fout("paranteze3.out");

int n;
char s[256];
char stiva[256];

int main()
{
```

```
    fin >> n;
    for(int k = 1 ; k <= n ; k ++)
    {
        fin >> s;
        int ok = 1;
        int nivel = 0;
        for(int i = 0 ; s[i] && ok ; i ++)
            if(s[i] == '(')
                stiva[++nivel]='(';
            else
                if(s[i] == '[')
                    stiva[++nivel]='[';
                else
                    if(nivel > 0)
                    {
                        if(s[i] == ')')
                            if(stiva[nivel] == '(')
                                nivel --;
                            else
                                ok = 0;
                        else
                            if(stiva[nivel] == '[')
                                nivel --;
                            else
                                ok = 0;
                    }
                    else
                        ok = 0;
        if(nivel > 0)
            ok = 0;
        fout << ok << "\n";
    }
}
```

#849 Paranteze2

CERINȚA
Se dă un șir de paranteze rotunde care se închid corect (corect parantezat). Să se determine adâncimea parantezării.
Pentru un șir de paranteze închise corect S, adâncimea parantezării, D(S) este definită astfel:
dacă șirul S este vid, D(S)=0
dacă S=(T), unde T este un șir de paranteze corect, D(S)=1+D(T)
dacă S=AB, unde A și B sunt șiruri de paranteze corecte, D(S)=max{D(A),D(B)}
DATE DE INTRARE
Fișierul de intrare paranteze2.in conține pe prima un șir de paranteze rotunde, corect parantezat.
DATE DE IEȘIRE
Fișierul de ieșire paranteze2.out va conține pe prima linie un număr M, reprezentând adâncimea parantezării.

RESTRICȚII ȘI PRECIZĂRI
șirul va conține cel mult 254 de paranteze

Exemplu
paranteze2.in
()((()())())
paranteze2.out
3
REZOLVARE

```
#include <fstream>

using namespace std;

ifstream cin ("paranteze2.in");
ofstream cout ("paranteze2.out");
char t[1000];
int k, km;
int main ()
{
cin>>t;
for(int i=0; t[i]=='(' || t[i]==')'; i++)
{
    if(t[i]=='(')
        k++, km=max(k, km);
    if(t[i]==')')
        k--, km=max(k, km);
}
cout<<km;
    return 0;
}
```

#2645 minlex

Se consideră un cuvânt format numai din litere mici și un număr natural nenul K.

CERINȚA
Să se determine cuvântul minim lexicografic obținut prin eliminarea a exact K litere din cuvântul inițial.

DATE DE INTRARE
Programul citește de la tastatură numărul K, apoi cuvântul.

DATE DE IEȘIRE
Programul va afișa pe ecran cuvântul rămas după eliminarea a exact K litere, minim lexicografic.

RESTRICȚII ȘI PRECIZĂRI
3 ≤ lungimea cuvântului ≤ 1.000.000
Cuvântul este format numai din litere mici ale alfabetului englez.
1 ≤ K < lungimea cuvântului
Literele rămase după eliminare nu-și pot schimba ordinea în cuvânt.

Exemplu
Intrare
5 zyizxtnfo
Ieșire
info

REZOLVARE

```
#include <bits/stdc++.h>
#define nmax 1000005
using namespace std;

char a[nmax];
int k;
char s[nmax];

int main()
{
    int i, top;
    char x;
    cin >> k;
    cin >> (s + 1);
    a[0] = 'A';
    top = 0;
    for (i = 1; s[i] && k > 0; ++i)
    {
        x = s[i];
        while (k > 0 && a[top] > x)
        {
            k--;
            top--;
        }
    a[++top] = x;
    }
    for( ; s[i]; i++)
        a[++top] = s[i];
    while (k > 0)
    {
        k--;
        top--;
    }
    a[top + 1] = 0;
    cout << (a + 1) << "\n";
    return 0;
}
```

#1884 UEMM1

CERINȚA
Se dă un șir cu n elemente, numere naturale. Să se afișeze, pentru fiecare element din șir, valoarea din șir aflată după acesta și mai mare decât acesta (Următorul Element Mai Mare). Dacă o asemenea valoare nu există, se va afișa -1.

DATE DE INTRARE
Programul citește de la tastatură numărul n, iar apoi cele n elemente ale șirului.
DATE DE IEȘIRE
Programul va afișa pe ecran cele n valori determinate, separate prin câte un spațiu.
RESTRICȚII ȘI PRECIZĂRI
$1 \leq n \leq 100.000$
elementele șirului vor fi mai mai mici decât 1.000.000

Exemplu
Intrare
5
3 4 3 5 1
Ieșire
4 5 5 -1 -1

REZOLVARE

```
#include<iostream>
#include<algorithm>
#include<cstring>
#include<cmath>
#include<cstdlib>

using namespace std;

#define NN 100001

int main()
 {
    int n  , v[NN], rez[NN];
    int s[NN], ns = 0;
    cin >> n;
    v[n + 1] = -1;
    for(int i = 1 ; i <= n ; i ++)
        cin >> v[i];
    s[ns = 1] = 1;
    for(int i = 2; i <= n ; i ++)
    {
        while(ns > 0 && v[i] > v[s[ns]])
        {
            rez[s[ns--]] = v[i];
        }
        s[++ns] = i;
    }
    while(ns > 0)
    {
        rez[s[ns--]] = -1;
    }
    for(int i = 1; i <= n ; i ++)
        cout << rez[i] << " ";
    return 0;
}
```