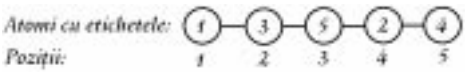
PROBLEME propuse pentru REZOLVARE

Vă prezentăm enunțurile celor șase probleme propuse spre rezolvare concurenților de la Olimpiada de Informatică a Europei Centrale, desfășurată în perioada 10-17 august 2001 în orașul Zalaegerszeg din Ungaria.

P070101: Lanț de atomi

Oamenii de știință studiază un tip special de molecule. Se știe că o moleculă este formată din *N* atomi diferiți, numerotați de la 1 la *N* și că molecula are o structură liniară. Oamenii de știință au în dotare un dispozitiv pentru măsurarea distanțelor. La o măsurătoare, instrumentul poate determina distanța dintre doi atomi ai moleculei. Distanța determinată de instrument indică valoarea absolută a diferenței dintre pozițiile în lanț ale celor doi atomi. Dacă nu se accesează nici un atom de mai mult de trei ori, molecula nu suferă nici o deteriorare. Dacă unul sau mai mulți atomi sunt accesați de patru ori, molecula va suferi o mică deteriorare. Dacă unul sau mai mulți atomi sunt accesați de cinci sau mai multe ori, molecula este distrusă.

Sarcina voastră este de a scrie un program care descoperă structura completă a moleculei, determinând ordinea atomilor din moleculă.



Biblioteca

Pentru a utiliza dispozitivul de măsurare a distanțelor aveți la dispoziție o bibliotecă cu numele `meter` care poate efectua trei operații (funcții):

- `Size` - trebuie apelată o singură dată la început, nu are argumente și returnează valoarea *N*. `Size` trebuie apelată înaintea oricărui apel al funcției `Span`.
- `Span` - este apelată cu doi parametri (etichetele a doi atomi) și returnează distanța dintre cei doi atomi.
- `Answer` - trebuie apelată la sfârșit și este folosită pentru a trimite soluția voastră. `Answer` are doi parametri *i* și *x* (numere întregi) și specifică faptul că eticheta atomului de pe poziția *i* a moleculei este *x*. Pentru fiecare *i* ($1 \le i \le N$), `Answer` trebuie apelată exact o dată în ordinea *crescătoare* a valorii *i*. Există întotdeauna două soluții simetrice; puteți determina doar una dintre ele.

Dialogul dintre programul vostru și bibliotecă va fi descris automat în fișierul text **`chain.out`**. Acest fișier este generat în urma apelurilor subprogramelor din bibliotecă și va conține, de asemenea, răspunsul vostru, numărul maxim de accesări ale atomilor și eventualele mesaje de eroare.

Atenție!!!

Nu programul vostru trebuie să creeze acest fișier; el este generat de bibliotecă.

Instrucțiuni pentru programatorii în Pascal

- În codul sursă folosiți instrucțiunea de includere **`uses`** `meter;`.

Instrucțiuni pentru programatorii în C/C++

- În codul sursă folosiți instrucțiunea `#include "meter.h";`.
- Creați în directorul problemei un fișier proiect cu numele **`chain.gpr`**, adăugați fișierele **`chain.c`** (**`chain.cpp`**) și **`meter.o`** în acest proiect și apoi *compilați* și/sau *executați* programul.

Experimentare

Puteți realiza experimente cu biblioteca, creând un fișier text cu numele **`chain.in`**. Fișierul va trebui să conțină două linii. Prima dintre acestea va conține numărul întreg *N* care reprezintă numărul atomilor din moleculă. A doua linie va conține un set de date de test: o secvență de etichete care constă în *N* numere întregi distincte cu valori cuprinse între 1 și *N*.

Exemplu pentru experimentare

`chain.in`

```
5
1 3 5 2 4
```

chain.out

```
Size=5
Span(1,2)=3
Span(1,3)=1
Span(2,3)=2
Span(4,5)=2
Span(1,4)=4
Span(3,5)=1
Your answer:
1 3 5 2 4
Max. Access/Atom:
3
```

Restricții

- Pentru numărul de atomi N, avem $5 \le N \le 10000$.
- Accesarea de către funcția `Span` a unui atom de mai mult de patru ori va întrerupe execuția programului vostru.
- Programul nu va trebui să citească date din nici un fișier sau să scrie date într-un anumit fișier.
- Pentru etichetele atomilor (x) și pozițiile atomilor (i), avem $1 \le i, x \le N$.
- Denumirile fișierelor bibliotecii pentru limbajul *FreePascal* sunt: **meter.ppw** și **meter.ow**.
- Declarațiile subprogramelor *Pascal* sunt:

```
function Size:Integer;
function Span(x,y:Integer):Integer;
procedure Answer(i,x:Integer);
```

- Denumirile fișierelor bibliotecii pentru limbajul *C/C++* sunt **meter.h** și **meter.o**.
- Declarațiile funcțiilor *C/C++* sunt:

```
int Size(void);
int Span(int x,int y);
void Answer(int i,int x);
```

Punctare

Dacă răspunsul vostru este corect și nici un atom nu a fost accesat de mai mult de trei ori, veți primi întregul punctaj. Dacă răspunsul vostru este corect, dar unul sau mai mulți atomi au fost accesați de patru ori, veți obține 50% din punctaj. Dacă răspunsul vostru este incorect sau un anumit atom a fost accesat de cinci ori, nu veți primi nici un punct.

Precizări

- Timpul de execuție este de 1 secundă/test.
- Limita de memorie este de 32 MB.
- Limita pentru dimensiunea sursei este de 1 MB.
- Opțiunile de compilare pentru *FreePascal* sunt:

```
-So -O2 -XS.
```

- Opțiunile de compilare pentru *C/C++* sunt:

```
-O2 -static meter.o.
```

P070102: Circuit imprimat

Un *circuit imprimat* este o placă formată din *noduri* și *segmente corespunzătoare firelor* care leagă perechi de noduri.

Considerăm un circuit imprimat special în care nodurile sunt aranjate pe un caroiaj dreptunghiular și firele leagă (pe verticală sau orizontală) doar noduri adiacente. Un circuit imprimat este *conex* dacă oricare două noduri distincte sunt legate prin cel puțin o secvență de fire. Se dă un circuit imprimat în care există deja câteva fire care leagă câteva noduri adiacente. Trebuie adăugate fire noi astfel încât circuitul imprimat să devină conex. Costul unui fir este `1` dacă este vertical și `2` dacă este orizontal.

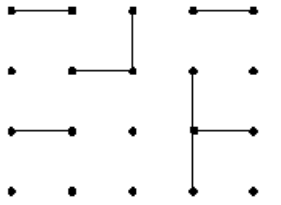


Figura 1

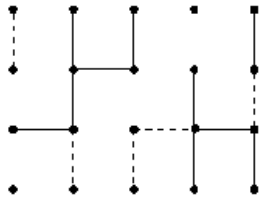


Figura 2

Trebuie să scrieți un program care determină un cost minim pentru obținerea unui circuit conex. Programul vostru va trebui să rezolve următoarele trei subprobleme:

A. Determinarea numărului de fire noi adăugate pentru obținerea unui circuit conex de cost minim.
B. Calcularea valorii costului minim.
C. Determinarea unei liste de fire pentru obținerea circuitului conex de cost minim.

Intrare

Prima linie a fișierului **circuit.in** conține două numere întregi, N și M. N ($1 \le N \le 100$) este numărul de linii și M ($1 \le M \le 100$) este numărul de coloane ale caroiajului. Nodurile circuitului sunt identificate prin coordonatele lor; nodul din colțul stânga-sus are coordonatele (1, 1), iar nodul din colțul dreapta-jos are coordonatele (N, M). Fiecare dintre următoarele N linii conține M numere întregi. Numărul din coloana j a liniei i descrie firele dintre perechile de noduri (i, j) și $(i + 1, j)$, și, de asemenea, dintre perechile de noduri (i, j) și $(i, j + 1)$ astfel:

- valoarea `0` indică faptul că nici nodurile din perechea (i, j) cu $(i + 1, j)$ și nici nodurile din perechea (i, j) cu $(i, j + 1)$ nu sunt legate prin nici un fir.
- valoarea `1` indică faptul că doar nodurile din perechea (i, j) cu $(i + 1, j)$ sunt legate printr-un fir.
- valoarea `2` indică faptul că doar nodurile din perchea (i, j) cu $(i, j + 1)$ sunt legate printr-un fir.
- valoarea `3` indică faptul că atât nodurile din perechea (i, j) cu $(i + 1, j)$, cât și nodurile din perechea (i, j) cu $(i, j + 1)$ sunt legate prin câte un fir.

Există anumite poziții în care nu toate cele patru valori posibile sunt valide (de exemplu, în poziția (N, M) doar valoarea `0` este validă).

Ieșire

Prima linie a fișierului **circuit.out** trebuie să conțină două numere întregi, K și V. K este numărul de fire noi care trebuie adăugate pentru obținerea unui circuit imprimat

conex de cost minim. *V* este valoarea costului minim. Următoarele *K* linii reprezintă lista firelor care trebuie adăugate pentru obținerea circuitului imprimat conex de cost minim. Fiecare linie va conține trei numere întregi (*i*, *j* și *d*) care vor descrie exact un fir nou. (*i*, *j*) sunt coordonatele unui nod, iar *d* poate avea fie valoarea `1`, fie valoarea `2`.

Valoarea `1` indică faptul că noul fir leagă nodurile (*i*, *j*) și (*i* + 1, *j*), iar valoarea `2` indică faptul că noul fir leagă nodurile (*i*, *j*) și (*i*, *j* + 1).

Exemplu

Fișierul de intrare din exemplu corespunde circuitului din Figura 1. Fișierul de ieșire reprezintă o modalitate posibilă de a obține circuitul imprimat conex de cost minim și este prezentat în Figura 2.

circuit.in

```
4 5
2 1 1 2 1
0 3 0 1 0
3 0 0 3 1
0 2 0 2 0
```

circuit.out

```
5 6
1 1 1
3 2 1
3 3 1
3 3 2
2 5 1
```

Punctare

Dacă răspunsul vostru este corect pentru subpunctul **A**. veți obține 1 punct. Veți obține încă două puncte dacă răspunsul este corect ***și*** pentru subpunctul **B**. Dacă răspunsul este corect pentru toate cele trei subpuncte, veți obține 5 puncte.

Pentru a obține punctele corespunzătoare subpunctelor **A**. și **B**. nu este necesar ca fișierul de ieșire să conțină date pentru subpunctul **C**.

Precizări

- Timpul de execuție este de 1 secundă/test.
- Limita de memorie este de 32 MB.
- Limita pentru dimensiunea sursei este de 1 MB.
- Opțiunile de compilare pentru *FreePascal* sunt:
  `-So -O2 -XS`.
- Opțiunile de compilare pentru C/C++ sunt:
  `-O2 -static`.

P070103: Călătorie în circuit

Echipa ta decide ca după terminarea competiției să facă o excursie în circuit prin țara gazdă. Doriți să vă deplasați într-un oraș destinație și să vă întoarceți în orașul de plecare. Singura cerință pentru echipa voastră este ca traseul de plecare și traseul de întoarcere să conțină cel mai mic număr posibil de străzi comune. (Un traseu nu poate să conțină aceeași stradă de două sau mai multe ori.)

Trebuie să scrieți un program care determină două trasee între orașul de plecare și orașul destinație, astfel încât numărul de străzi comune ale celor două trasee să fie cât mai mic posibil.

Intrare

Prima linie din fișierul **trip.in** conține două numere întregi, *S* și *D* (*S* ≠ *D*) care reprezintă eticheta orașului de plecare și respectiv a orașului destinație. A doua linie conține două numere întregi, *N* și *M*, unde *N* (3 ≤ *N* ≤ 1000) este numărul de orașe și *M* (2 ≤ *M* ≤ 100000) este numărul de străzi care leagă orașele. Orașele sunt etichetate cu numere naturale cuprinse între 1 și *N*. Fiecare dintre următoarele *M* linii din fișier conține două numere întregi, *P* și *Q* (1 ≤ *P*, *Q* ≤ *N*, *P* ≠ *Q*), având semnificația că există o stradă cu sens dublu de circulație între orașele *P* și *Q*. Există cel mult o stradă între oricare două orașe.

Ieșire

Prima linie a fișierului **trip.out** trebuie să conțină un număr întreg reprezentând cel mai mic număr posibil de străzi comune ale celor două trasee. A doua linie trebuie să conțină traseul de plecare sub forma unei secvențe de etichete ale orașelor, inclusiv orașul de plecare și cel destinație. A treia linie trebuie să conțină traseul de întoarcere, sub forma unei secvențe de etichete ale orașelor, inclusiv orașul destinație și cel de plecare. Dacă există mai multe perechi de trasee având același număr minim de străzi comune, programul vostru poate să furnizeze la ieșire oricare dintre acestea. Dacă nu există nici un traseu de la orașul de plecare la orașul destinație, atunci prima și singura linie a fișierului de ieșire trebuie să conțină doar valoarea `-1`.

Exemplu

trip.in

```
1 6
7 8
2 1
1 3
2 3
4 2
4 5
5 6
7 5
6 7
```

trip.out

```
2
1 3 2 4 5 7 6
6 5 4 2 1
```

Punctare

Dacă prima linie a fișierului de ieșire conține răspunsul corect, veți obține 2 puncte. Dacă prima linie conține soluția

corectă și a doua și a treia linie conțin trasee corecte, veți obține încă 3 puncte.

Precizări

- Timpul de execuție este de 1 secundă/test.
- Limita de memorie este de 32 MB.
- Limita pentru dimensiunea sursei este de 1 MB.
- Opțiunile de compilare pentru *FreePascal* sunt:
  `-So -O2 -XS`.
- Opțiunile de compilare pentru *C/C++* sunt:
  `-O2 -static`.

P070104: Hartă de biți

Pozele alb-negru sunt deseori stocate sub forma unor hărți de biți. O *hartă de biți* este un caroiaj dreptunghiular de pixeli.

O *linie poligonală* între pixelii P și Q este o secvență de pixeli negri $P = P_1, P_2, \ldots, P_k = Q$, unde P_i și P_{i+1} ($i = 1, \ldots, k - 1$) sunt pixeli adiacenți (pe orizontală sau verticală, dar nu pe diagonală). O linie poligonală $P_1, P_2, \ldots, P_k$ este *închisă* dacă $P_1 = P_k$, și $P_i \neq P_j$ ($i = 1, \ldots, k - 1$, $j = 2, \ldots, k$) pentru $i < j$ cu excepția cazului în care $i = 1$ și $j = k$ (adică linia poligonală nu conține același pixel de două ori).

O *mulțime de pixeli negri* S este *conexă* dacă pentru fiecare pereche de pixeli (P, Q) din S există cel puțin o linie poligonală L între P și Q astfel încât toți pixelii de pe linia poligonală L fac parte din mulțimea S.

O *componentă* a unei hărți de biți este o mulțime conexă maximală de biți negri. O componetă poate conține în interior *goluri*. Un gol este format din pixeli albi care se află în interiorul unei linii poligonale închise. O *componentă compactă* nu conține în interior ***nici un gol***.

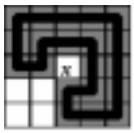


Figura 1

Atenție!!!

Pixelul alb marcat cu *x* aflat în mijlocul Figurii 1 nu se află în interiorul liniei poligonale marcate.

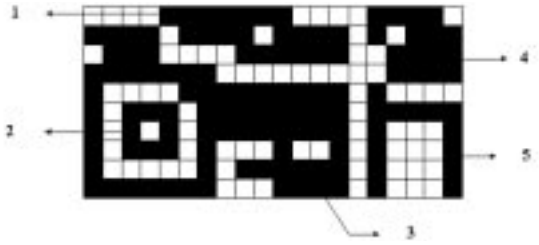


Figura 2

În Figura 2 se află o hartă de biți cu cinci componente, dintre care două sunt compacte.

Scrieți un program care determină numărul total de componente și numărul de componente compacte ale unei hărți de biți.

Codificare

Hărțile de biți investigate sunt codificate (comprimate, arhivate), folosind metoda descrisă în continuare.

Fiecare linie este codificată printr-o secvență de numere întregi $W_1, B_1, \ldots, W_k, B_k$, unde W_i este numărul de pixeli albi consecutivi, iar B_i este numărul de pixeli negri consecutivi. De exemplu, codificarea primei linii a hărții de biți din Figura 2 este secvența `4 7 4 4 1 0`.

Componentele 4 și 5 sunt compacte, în timp ce componentele 1, 2 și 3 nu sunt compacte.

Intrare

Prima linie a fisierului **`bitmap.in`** conține două numere întregi pozitive, N și M. N ($2 \leq N \leq 10000$) reprezintă numărul de linii, iar M ($2 \leq M \leq 1000$) reprezintă numărul de coloane ale hărții de biți. Următoarele N linii conțin harta de biți codificată așa cum s-a descris în paragraful referitor la *codificare*. Fiecare linie se termină cu valoarea `-1`.

Ieșire

Fișierul **`bitmap.out`** trebuie să conțină două linii, cu un singur număr întreg pe fiecare linie. Prima linie conține numărul tuturor componentelor, iar a doua linie conține numărul componentelor compacte din harta de biți de la intrare.

Exemplu

`bitmap.in`

```
10 20
4 7 4 4 1 0 -1
0 4 1 4 1 4 1 1 1 3 -1
1 3 4 6 2 4 -1
0 7 9 4 -1
0 1 4 9 1 1 4 0 -1
0 1 1 3 1 8 1 5 -1
0 1 1 1 1 1 1 8 1 1 3 1 -1
0 1 1 3 1 1 3 1 2 1 1 1 3 1 -1
0 1 5 1 1 6 1 1 3 1 -1
0 7 3 4 1 1 3 1 -1
```

`bitmap.out`

```
5
2
```

Punctare

Dacă prima linie a fișierului vostru de ieșire conține numărul corect al tuturor componentelor, veți obține 50% din punctaj. Dacă a doua linie a fișierului vostru de ieșire conține numărul corect al componentelor compacte, veți obține ***încă*** 50% din punctaj.

Precizări

- Timpul de execuție este de 10 secunde/test.
- Limita de memorie este de 32 MB.
- Limita pentru dimensiunea sursei este de 1 MB.
- Opțiunile de compilare pentru *FreePascal* sunt:
  `-So -O2 -XS`.
- Opțiunile de compilare pentru *C/C++* sunt:
  `-O2 -static`.

P070105: Șabloane Wildcard

Șabloanele wildcard sunt folosite frecvent pentru a specifica mulțimi de nume. De exemplu, putem specifica toate numele de fișiere care încep cu `h` și se termină cu `bak` prin șablonul `h*bak`.

Un *șablon wildcard* este un string care poate conține caractere `*` considerate *wildcard*-uri. Un string W se potrivește peste un șablon P dacă W poate fi obținut din P înlocuind fiecare caracter `*` din P cu câte un string (eventual unul vid). (Stringuri diferite pot fi folosite pentru aparițiile diferite ale caracterului `*`.) Pentru o pereche de șabloane P_1 și P_2, Q este un *șablon comun* al șabloanelor P_1 și P_2 dacă orice string care se potrivește peste Q se potrivește atât peste P_1 cât și peste P_2.

O mulțime $Q_1, Q_2, ..., Q_L$ de șabloane comune este numită *completă* dacă orice string care se potrivește atât peste P_1 cât și peste P_2 se potrivește peste cel puțin un element Q_i al mulțimii.

Trebuie să scrieți un program care, pentru o pereche dată de șabloane P_1 și P_2 determină:

A. cel puțin un șablon comun (dar nu și șabloane incorecte); sau
B. o mulțime completă de șabloane care nu conține mai mult de 6666 de elemente; sau
C. o mulțime completă de șabloane cu număr minim de elemente (adică nu există nici o altă mulțime completă de șabloane care să aibă mai puține elemente).

Atenție!!!

Dacă rezolvați subpunctul C. veți obține un ***bonus*** de 50% la punctaj.

Intrare

Prima și a doua linie a fișierului **`pattern.in`** conține șabloanele P_1 și P_2.

Șabloanele sunt formate din litere mici cuprinse între `'a'` și `'z'` și caracterul `'*'`. Fiecare șablon conține cel mult 20 de caractere. Numărul de apariții ale caracterului `*` în fiecare șablon este cuprins între 0 și 6.

Ieșire

Prima linie a fișierului **`pattern.out`** trebuie să conțină un număr întreg K, care reprezintă numărul șabloanelor care formează soluția. Următoarele K linii conțin câte un șablon comun; acestea sunt șabloanele care formează soluția problemei.

Ordinea șabloanelor comune nu are importanță. Atât subpunctul B. cât și subpunctul C. pot fi rezolvate în limita specificată de 6666 de elemente. Dacă nu există nici un șablon comun al șabloanelor P_1 și P_2, atunci prima și singura linie a fișierului va conține valoarea 0.

Exemplu

`pattern.in`

```
*ab*
ba*b
```

`pattern.out` (pentru subpunctul C.)

```
4
ba*ab*b
bab*b
ba*ab
bab
```

Punctare

Dacă soluția voastră rezolvă subpunctul A. veți obține 5 puncte. Dacă soluția voastră rezolvă subpunctul B. veți obține 10 puncte. Dacă soluția voastră rezolvă subpunctul C. veți obține 15 puncte.

Precizări

- Timpul de execuție este de 7 secunde/test.
- Limita de memorie este de 32 MB.
- Limita pentru dimensiunea sursei este de 1 MB.
- Opțiunile de compilare pentru *FreePascal* sunt:
  `-So -O2 -XS`.
- Opțiunile de compilare pentru *C/C++* sunt:
  `-O2 -static`.

P070106: Grup majoritar

Copiii dintr-o școală fac parte din două grupuri disjuncte. Se știe că unul din cele două grupuri, numit grup majoritar conține mai mulți copii decât celălalt. Se urmărește determinarea unui copil care aparține grupului majoritar. Singura operație care se poate efectua este de a întreba dacă doi copii oarecare aparțin sau nu aceluiași grup.

Scrieți un program care determină un copil care aparține grupului majoritar, punând *cât mai puține întrebări*.

Biblioteca

Pentru a putea pune întrebări, aveți la dispoziție o bibliotecă `query` care conține trei subprograme:

- `Size` - trebuie apelat o singură dată, la începutul programului, fără parametri; acesta va returna numărul N, reprezentând numărul de copii. Copiii sunt numerotați de la 1 la N. `Size` trebuie apelat înainte de primul apel al subprogramului `Member`.
- `Member` - are doi parametri, reprezentând numerele de ordine a doi copii; dacă cei doi copii aparțin aceluiași grup, subprogramul returnează `1`, altfel returnează `0`.
- `Answer` - se apelează o singură dată la sfârșitul programului și trebuie folosit pentru a transmite soluția voastră. `Answer` are un singur parametru întreg, reprezentând numărul de ordine al copilului care aparține grupului majoritar.

Se ține evidența dialogului dintre programul vostru și bibliotecă și se salvează în fișierul text **`select.out`**. Acest fișier conține, de asemenea, răspunsul vostru și precizarea dacă acest răspuns este corect sau nu.

Atenție!!!

Nu există teste predefinite, biblioteca produce răspunsuri "din mers". Dacă încercați să dați un răspuns înainte să fiți

100% siguri de corectitudinea lui, primiți 0 puncte; așa că nu încercați să dați cu banul! Răspunsul vostru este acceptat numai dacă pentru oricare două grupuri disjuncte, care corespund răspunsurilor primite la întrebările puse de program, copilul dat de programul vostru ca fiind în grupul majoritar, se află într-adevăr acolo. Biblioteca forțează programul vostru să pună toate întrebările care sunt necesare pentru a identifica un membru al grupului majoritar.

Instrucțiuni pentru programatorii în Pascal

- folosiți instrucțiunea de includere **uses** `query;` în codul sursă.

Instrucțiuni pentru programatorii în C/C++

- folosiți instrucțiunea `#include "query.h";` în codul sursă;
- creați în directorul problemei un fișier proiect cu numele **select.gpr**, adăugați fișierele **select.c** (**chain.cpp**) și **select.o** în acest proiect și apoi *compilați* și/sau *executați* programul.

Experimentarea

Puteți face experimente cu biblioteca creând un fișier text **select.in**. Prima și singura linie trebuie să conțină numărul de copii *N*. *N* trebuie să fie impar!

Exemplu

select.in

```
7
```

select.out

```
Size=7
Member(1,2)=0
Member(3,4)=1
Member(5,6)=1
Member(4,6)=0
Your Answer: 7, Correct
Majority Group:
2 5..7
Non-majority Group:
1 3 4
Number of Queries: 4
Full Possible Score: 3
Your Score: 3
```

De exemplu, răspunsul `1` nu este acceptat, deoarece, cum toate răspunsurile dau același rezultat pentru grupurile {`2`, `5`, `6`, `7`} și {`1`, `3`, `4`}, `1` nu e membru în grupul majoritar {`2`, `5`, `6`, `7`}.

Restricții

- $5 \le N < 30000$, N impar.
- Programul vostru nu trebuie să citească date dintr-un fișier sau să scrie date într-un fișier.
- Copiii sunt identificați prin numerele de ordine i, unde $1 \le i \le N$.
- Denumirile fișierelor bibliotecii pentru limbajul *FreePascal* sunt: **query.ppw** și **query.ow**.
- Declarațiile subprogramelor *Pascal*:

```
function Size:Integer;
function Member(x,y:Integer):Integer;
procedure Answer(x:Integer);
```

- Denumirile fișierelor bibliotecii pentru limbajul *C/C++* sunt: **query.h** și **query.o**
- Declarațiile funcțiilor *C/C++*:

```
int Size(void);
int Member(int x,int y);
void Answer(int x);
```

Punctare

Dacă răspunsul vostru este corect și numărul de întrebări pe care programul le-a pus este K, atunci obțineți *max*(0, $N - K$) puncte, unde N este numărul de copii.

Precizări

- Timpul de execuție este de 1 secundă/test.
- Limita de memorie este de 32 MB.
- Limita pentru dimensiunea sursei este de 1 MB.
- Opțiunile de compilare pentru *FreePascal* sunt:
  `-So -O2 -XS`.
- Opțiunile de compilare pentru *C/C++* sunt:
  `-O2 -static query.o`.

România la CEOI 2001

La CEOI 2001, România a obținut două medalii de argint (prin Daniel Dumitran din București și Florin Ghețu din Focșani) și una de bronz (prin Victor Costan).

Vencel Bors din Oradea, a fost foarte aproape de a obține o medalie.

Delegația României l-a avut ca team leader pe dl. lect. univ. Stelian Ciurea și ca deputy leader pe dl. prof. Doru Popescu Anastasiu.

Au mai participat, în calitate de invitați, d-na lect. univ. Clara Ionescu, Mihai Scorțaru și Beatrice Ionescu.