IOI 2003

În perioada 16-23 august 2003 a avut loc cea de-a XV-a ediție a concursului internațional de programare IOI. Acesta a fost găzduit de Universitatea Wisconsin Parkside, din Statele Unite ale Americii.

Ediția de anul acesta a *Olimpiadei Internaționale de Informatică* a fost găzduită de *Universitatea Wisconsin Parkside*, situată în orașul american *Kenosha*.

Sponsorul oficial al acestei competiții a fost compania *Microsoft*.

În acest an echipa României a fost reprezentată de patru concurenți. Aceștia sunt:

Victor-Marius Costan, București
Radu Berinde, București
Dan Ghinea, București
Ștefan Ciobâcă, Suceava

Victor-Marius Costan și *Radu Berinde*, obținând 424 de puncte, respectiv 353,9 puncte, au câștigat medalii de aur. Ceilalți doi concurenți români, *Ștefan Ciobâcă* și *Dan Ghinea,* obținând 289,9 puncte, respectiv 366,4 puncte, au câștigat medalii de argint.

Cu aceste rezultate obținute de echipa țării noastre, România a ocupat locul I în clasamentul pe echipe, alături de *Statele Unite ale Americii*, *Coreea* și *Suedia*.

Nici unul dintre participanții la ediția din acest an a *Olimpiadei Internaționale de Informatică* nu a reușit să obțină punctajul maxim, și anume 600.

Cel mai mare număr de puncte a fost obținut de către concurentul coreean *Hwan-Seung Yeo* care a acumulat cu rezolvările sale 455,4 puncte, aflându-se la o diferență de aproximativ 20 de puncte de următorul.

Concurenților li s-au pus la dispoziție sisteme dotate cu procesoare *Intel Pentium* 4 cu o frecvență de 2,2 *GHZ*, 256 MB memorie RAM și unități *CD-RW* pe care au fost instalate sistemele de operare *Microsoft Windows XP Professional* și *Linux RedHat* 9. În timpul concursului, participanților li s-a permis să folosească propriile tastaturi, dispozitive de indicare (de exemplu, alt *mouse*) și propriile *CD*-uri pentru a face copii de siguranță.

În timpul evaluării programele concurenților au avut la dipoziție 64MB de memorie *RAM.*

În continuare vă prezentăm enunțurile celor șase probleme propuse spre rezolvare.

P060301: Comparări de cod

Compania *Racine Business Networks* (*RBN*) a dat în judecată compania *Heuristic Algorithm Languages* (*HAL*), invocând faptul că *HAL* a preluat o parte din codul sursă *RBN UNIX™* și l-a inclus în sistemul de operare cu fișierele sursă deschise *HALnix*.

Ambele companii *RBN* și *HAL* utilizează un limbaj de programare cu câte o instrucțiune pe linie de forma: `STOREA = STOREB + STOREC`, unde `STOREA`, `STOREB` și `STOREC` sunt nume de variabile.

Primul nume de variabilă începe din prima coloană, după el urmează un spațiu, semnul egal, un alt spațiu, al doilea nume de variabilă, iarăși un spațiu, simbolul plus, din nou un spațiu și al treilea nume de variabilă. Același nume de variabilă poate apărea pe o singură linie mai mult decât o singură dată. Numele de variabile sunt formate din cel puțin una și cel mult 8 litere mari ale alfabetului englez.

Compania *RBN* pretinde că *HAL* a copiat direct din codul sursă elaborat de *RBN* o anumită secvență de linii consecutive, făcând numai modificări minore:

- Compania *RBN* pretinde că în scopul tăinuirii infracțiunilor sale, compania *HAL* a schimbat unele nume de variabile. Mai exact, compania *HAL* a preluat o secvență de linii din programul elaborat de compania *RBN* și pentru fiecare variabilă din această secvență a schimbat toate aparițiile numelui respectiv cu un nume nou, uneori numele nou putând să coincidă cu cel inițial. Desigur, variabilele distincte din programul inițial au denumiri distincte și în programul final.
- De asemenea, compania *RBN* pretinde că *HAL* e posibil să fi modificat ordinea de apariție a variabilelor din partea dreaptă a semnului egal din unele linii, de exemplu, linia `STOREA = STOREB + STOREC` posibil să fi fost modificată în `STOREA = STOREC + STOREB`.
- Compania *RBN* afirmă că *HAL* nu a modificat ordinea liniilor din codul sursă propus de *RBN*.

Elaborați un program, care, având ca date inițiale codurile sursă ale ambelor companii, găsește lungimea celei mai lungi secvențe de linii consecutive din programul elaborat de compania *HAL*, care poate fi obținută dintr-o secvență de linii consecutive din programul *RBN*, utilizând transformările descrise mai sus. Menționăm că secvențele de linii consecutive din cele două programe nu trebuie să înceapă în aceeași poziție în cadrul programelor.

Date de intrare
Fișierul de intrare **code.in** conține pe prima linie două numere întregi pozitive *R* și *H,* separate prin spațiu, care reprezintă numărul de linii de cod ale programului sursă *RBN*, respectiv numărul de linii de cod ale programului *HAL*. Pe următoarele *R* linii se află programul *RBN*, iar în continuare, pe următoarele *H* linii se află programul *HAL*.

Date de ieșire
Fișierul de ieșire **code.out** va conține un singur număr întreg pozitiv care reprezintă lungimea celei mai lungi secvențe de linii consecutive din programul *RBN* pe care compania *HAL* le-ar fi copiat și transformat.

Restricții
- $1 \le R \le 1000$;
- $1 \le H \le 1000$.

Exemplu
code.in
```
4 3
RA = RB + RC
RC = D + RE
RF = RF + RJ
RE = RF + RF
HD = HE + HF
HM = HN + D
HN = HA + HB
```

code.out
```
2
```

Explicație
Liniile 1 și 2 ale programului *RBN* conicid cu liniile 2 și 3 ale programului *HAL*, dacă numele de variabile sunt înlocuite după cum urmează: `RA` → `HM`, `RB` → `D`, `RC` → `HN`, `D` → `HA`, `RE` → `HB`. Nu există secvențe de trei sau mai multe linii care ar fi compatibile.

Timp de execuție: 2 secunde/test

P060302: Întreținerea potecilor
Vacile fermierului *John* doresc să umble libere pe cele *N* pășuni din fermă, numerotate de la 1 la *N*, chiar dacă pășunile sunt separate prin păduri. Vacile doresc să întrețină potecile existente între perechi de pășuni astfel încât să poată parcurge drumul care leagă oricare pășune de oricare alta, utilizând potecile întreținute. Vacile pot umbla pe o potecă întreținută în ambele sensuri.

Vacile nu construiesc poteci. În schimb, ele întrețin potecile animalelor sălbatice pe care le-au descoperit. În orice săptămână ele pot să aleagă să întrețină unele sau toate potecile pe care le cunosc.

Curios, la începutul fiecărei săptămâni, vacile descoperă o nouă potecă de animale sălbatice. După aceea trebuie să decidă mulțimea de poteci pe care le vor întreține în săptămâna respectivă astfel încât să poată parcurge drumul de la orice pășune la orice alta. La un moment dat, vacile nu pot folosi decât potecile pe care le întrețin în momentul respectiv.

Vacile doresc întotdeauna să minimizeze lungimea totală a potecilor pe care trebuie să le întrețină. Vacile pot să aleagă pentru întreținere orice submulțime de poteci de animale sălbatice pe care le cunosc, indiferent de potecile pe care le-au menținut în săptămâna precedentă.

Potecile de animale sălbatice (chiar și atunci când sunt întreținute) nu sunt niciodată drepte.

Două poteci care unesc aceleași două pășuni pot avea lungimi diferite. Chiar dacă două poteci se intersectează, vacile sunt atât de concentrate încât refuză să schimbe poteca, exceptând cazul în care sunt pe o pășune.

La începutul fiecărei săptămâni, vacile vă vor descrie poteca de animale sălbatice pe care au descoperit-o. Programul vostru trebuie să afișeze lungimea totală minimă a potecilor pe care vacile trebuie să le întrețină în săptămâna respectivă, astfel încât să poată parcurge drumurile care între oricare două pășuni, dacă există o astfel de mulțime de poteci.

Date de intrare
Datele se citesc de la intrarea standard. Prima linie va conține două numere întregi *N* și *W*, separate printr-un singur spațiu, care reprezintă numărul de pășuni din fermă, respectiv numărul de săptămâni pe care programul le va rezolva.

Pentru fiecare săptămână, datele se dau pe o singură linie care conține informații despre poteca de animale sălbatice descoperită. Această linie conține trei numere întregi separate prin spații: numerele de ordine ale celor două pășuni care reprezintă extremitățile potecii) și lungimea potecii.

Date de ieșire
Rezultatele se vor scrie la ieșirea standard. În momentul în care programul vostru află date despre noua potecă de animale sălbatice descoperită trebuie să afișeze o singură linie care conține lungimea totală minimă a potecilor pe care vacile trebuie să le întrețină astfel încât să poată parcurge drumurile care leagă oricare pășune la oricare alta.

Dacă nici o mulțime de poteci nu permite vacilor să traverseze drumurile între oricare două pășuni, programul va afișa `"-1"`.

Programul vostru trebuie să se termine după ce afișează răspunsul pentru ultima săptămână.

Restricții și precizări

- $1 \le N \le 200$;
- $1 \le W \le 6000$;
- lungimile potecilor sunt numere întregi cuprinse între 1 și 10000;
- nici o potecă de animale sălbatice nu are aceeași păsune în ambele extremități.

Exemplu

Intrare	*Ieșire*
`4 6`	
`1 2 10`	`-1`
`1 3 8`	`-1`
`3 2 3`	`-1`
`1 4 3`	`14`
`1 3 6`	`12`
`2 1 2`	`8`

Explicație

După descoperirea primei poteci, mulțimea de poteci va fi formată dintr-un singur element, deci nici o potecă din mulțime nu leagă păsunea 4 de restul păsunilor.

Nici după descoperirea celei de-a doua și a treia poteci, în mulțime nu va exista o potecă ce leagă păsunea 4 de restul păsunilor.

După descoperirea celei de-a patra poteci, vor trebui întreținute potecile: `1 4 3`, `1 3 8` și `3 2 3` a căror lungime totală este 14.

După descoperirea celei de-a cincia poteci, vor trebui întreținute potecile: `1 4 3`, `1 3 6` și `3 2 3` a căror lungime totală este 12.

După descoperirea ultimei poteci, vor trebui întreținute potecile: `1 4 3`, `2 1 2` și `3 2 3` a căror lungime totală este 8.

Timp de execuție: 1 secundă/test

P060303: Inversare

Să considerăm o mașină care execută două operații (pe scurt denumită *TOM*) și care are nouă regiștri, numerotați de la 1 la 9. Fiecare registru poate memora un număr natural din intervalul [0, 10000]. Mașina poate executa două operații:

- `S i j` - memorează în registrul `j` valoarea din registrul `i` la care se adaugă `1` și valoarea lui `i` poate fie egală cu valoarea lui `j`;
- `P i` - afișează valoarea memorată în registrul `i`.

Un program *TOM* include un set de valori inițiale ale regiștrilor și o secvență de operații. Fiind dat un număr natural *N*, generați un program *TOM* care să afișeze secvența descrescătoare de numere întregi *N*, *N* - 1, *N* - 2, ..., 0. Numărul maxim de operații `S` consecutive trebuie să fie cât mai mic posibil.

În continuare aveți un exemplu de program *TOM* și execuția lui pentru *N* = 2.

Operație	Valori regiștrilor	Valoarea afișată
Valori inițiale	`0 2 0 0 0 0 0 0 0`	
`P 2`	`0 2 0 0 0 0 0 0 0`	`2`
`S 1 3`	`0 2 1 0 0 0 0 0 0`	
`P 3`	`0 2 1 0 0 0 0 0 0`	`1`
`P 1`	`0 2 1 0 0 0 0 0 0`	`0`

Date de intrare

Aceasta este o problemă cu fișiere deschise. Fișierele de intrare sunt formate din două linii.

Prima linie a unui fișier de intrare conține un singur număr întreg *K*, reprezintă numărul testului, iar a doua linie conține numărul natural *N*.

La acest concurs au fost puse prin intermediul *server*-ului 16 fișiere de intrare.

Date de ieșire

Nu sunt restricții în ceea ce privește numele fișierelor de ieșire, dar acestea trebuie să respecte formatul descris în continuare.

Prima linie a unui fișier de ieșire trebuie să conțină șirul `"FILE reverse K"`, unde `K` reprezintă numărul testului și este cuprins între 1 și 16.

A doua linie trebuie să conțină 9 numere separate între ele prin spațiu care reprezintă valorile inițiale ale regiștrilor, pe care le doriți, în ordine, adică primul număr reprezintă valoarea inițială a primului registru, cel de-al doilea număr reprezintă valoarea inițială a celui de-a doilea registru etc.

Restul fișierului de ieșire trebuie să conțină o listă ordonată de operații care trebuie executate, câte o operație pe linie.

Astfel, a treia linie conține prima operație care trebuie executată, și așa mai departe. Ultima linie a fișierului trebuie să fie cea care afișează valoarea `0`.

Fiecare linie trebuie să conțină o operație validă. Operațiile trebuie să fie formatate ca în exemplu.

Restricții

- $0 \le N \le 255$;
- numărul de operații `S` consecutive nu trebuie să fie mai mare decât 131;
- o operație `S` nu trebuie să ducă la depășirea valorii maxime admise pentru un registru.

Exemplu

Conținut fișier de intrare

```
1
2
```

Conținut fișier de ieșire 1

```
FILE reverse 1
0 2 1 0 0 0 0 0 0
P 2
P 3
P 1
```

Conținut fișier de ieșire 2

```
FILE reverse 1
0 2 0 0 0 0 0 0 0
P 2
S 1 3
P 3
P 1
```

Modalitate de acordare a punctajului

Din punctajul maxim care putea fi obținut pentru un test s-a acordat 20% pentru corectitudinea fișierului de ieșire.

Un fișier de ieșire este corect dacă respectă formatul descris și, de asemenea, respectă restricțiile impuse.

Restul de 80% a fost acordat pentru optimalitatea programului *TOM* trimis.

Optimalitatea unui program *TOM* corect este măsurată de numărul maxim de operații `S` consecutive din program, care trebuie să fie cât mai mic posibil.

P060304: Frontiera

Fermierul *Don* se uită la gardul care înconjoară terenul său plat de formă pătrată de N metri pe N metri. Un colț al gardului este în origine (0, 0), iar colțul opus este în punctul de coordonate (N, N); laturile gardului fermierului *Don* sunt paralele cu axele X și Y.

Stâlpii gardului nu apar numai în cele patru colțuri, ci și la fiecare metru de-a lungul gardului, deci în total $4 \cdot N$ stâlpi. Stâlpii sunt verticali și considerați de rază neglijabilă. Fermierul *Don* vrea să determine câți stâlpi din gardul său poate vedea atunci când stă într-o poziție dată.

Terenul fermierului *Don* conține stânci uriașe care obturează vizibilitatea către anumiți stâlpi, deoarece el nu este suficient de înalt pentru a se uita peste aceste stânci.

Baza fiecărei stânci este un poligon convex cu arie nenulă ale cărui vârfuri sunt în puncte de coordonate întregi. Stâncile stau complet verticale. Stâncile nu se suprapun, nu ating alte stânci, nu îl ating pe fermierul *Don* și nici gardul.

Fermierul *Don* nu atinge gardul, nu stă într-o stâncă sau pe o stâncă.

Fiind dată dimensiunea gardului, poziția și forma stâncilor și poziția în care se află fermierul *Don*, determinați numărul stâncilor pe care fermierul *Don* le poate vedea.

Date de intrare

Prima linie a fișierului de intrare **`boundary.in`** conține două valori întregi N și R, separate printr-un singur spațiu, reprezentînd lungimea laturii terenului, respectiv numărul de stânci care obturează vizibilitatea către anumiți stâlpi.

Cea de-a doua linie conține două numere întregi separate printr-un singur spațiu care reprezintă coordonatele X și Y ale fermierului *Don* din interiorul gardului.

Restul fișierului de intrare descrie cele R stânci, astfel:

- descrierea stâncii i începe cu o linie conținând un singur număr întreg p_i, reprezentând numărul de vârfuri ale bazei stâncii.
- fiecare dintre următoarele p_i linii conține o pereche de numere întregi, separate prin spațiu, reprezentând coordonatele X și Y ale unui vârf. Vârfurile bazei unei stânci sunt distincte și sunt date în sensul invers al acelor de ceasornic.

Date de ieșire

Fișierul de ieșire **`boundary.out`** conține o singură linie pe care se va afla un singur număr întreg care reprezintă numărul de stâlpi pe care îi poate vedea fermierul *Don* din locul în care se află.

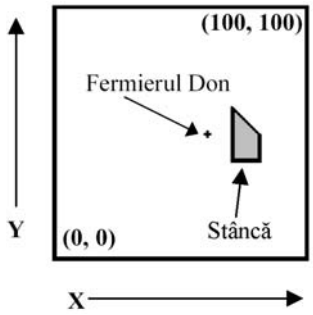
Restricții

- $2 \le N \le 500.000$;
- $1 \le R \le 30.000$;
- $3 \le p_i \le 20$, $1 \le i \le R$.

Exemplu

`boundary.in`

```
100 1
60 50
5
70 40
75 40
80 40
80 50
70 60
```



`boundary.out`

```
319
```

Timp de execuție: 1 secundă/test

P060305: Ghicește vaca

Cele N vaci din cireada fermierului *John* sunt foarte asemănătoare și sunt numerotate de la 1 la N. Când fermierul *John* duce o vacă la culcare în staulul ei, trebuie să determine despre care vacă este vorba ca să o pună în staulul corect.

Vacile sunt diferențiate utilizând P proprietăți, numerotate de la 1 la P, fiecare dintre ele având trei valori posibile. De exemplu, culoarea etichetei de pe urechea vacii poate fi *galben*, *verde* sau *roșu*. Pentru simplitate, valorile fiecărei proprietăți sunt desemnate prin literele `'X'`, `'Y'` și `'Z'`.

Oricare două vaci ale fermierului *John* diferă prin cel puțin o proprietate.

Scrieți un program care, plecând de la proprietățile vacilor din cireada fermierului *John*, determină care vacă trebuie dusă la culcare.

Programul poate pune fermierului *John* cel mult 100 de întrebări de forma:

"*Valoarea proprietății T a vacii aparține mulțimii S?*"

Încercați să puneți cât mai puține întrebări pentru a ghici care vacă va merge la culcare.

Date de intrare

Prima linie a fișierului de intrare **guess.in** conține două valori întregi N și P, separate printr-un singur spațiu, care reprezintă numărul de vaci, respectiv numărul de proprietăți.

Aceasta este o problemă interactivă.

Întrebările sunt puse prin intermediul ieșirii standard.

După punerea unei întrebări, programul trebuie să citească o singură linie care conține un singur număr întreg. Dacă numărul citit este egal cu 1 înseamnă că proprietatea specificată pentru vaca trimisă la culcare se află în mulțimea de valori dată, în caz contrar, dacă numărul citit este egal cu 0 înseamnă că proprietatea specificată nu se află în mulțimea de valori dată.

Date de ieșire

Programul trebuie să pună întrebări despre vaca trimisă la culcare scriind la ieșirea standard o linie care începe cu caracterul `'Q'` urmat de un spațiu, numărul proprietății, un spațiu și o mulțime de una sau mai multe valori separate prin spații.

De exemplu, `Q 1 Z Y` semnifică întrebarea "*Proprietatea 1 a vacii trimise la culcare este* `'Z'` *sau* `'Y'`*?*". Proprietatea trebuie să fie un număr întreg din intervalul [1, P].

Valorile trebuie să fie `'X'`, `'Y'` sau `'Z'` și nici o valoare nu trebuie să apară într-o întrebare de mai multe ori.

Ultima linie pe care o afișează programul trebuie să înceapă cu caracterul `'C'` urmat de un spațiu și apoi un singur număr întreg care specifică vaca pe care programul a determinat-o ca fiind cea pe care fermierul *John* o duce la culcare.

Restricții și precizări

- $1 \le N \le 50$;
- $1 \le P \le 8$;
- programul vostru nu trebuie să pună mai mult de 100 de întrebări.

Exemplu

guess.in

```
4 2
X Z
X Y
Y X
Y Y
```

Interactivitate
Întrebare: `Q 1 X Z`
Răspuns: `0`
Întrebare: `Q 2 Y`
Răspuns: `1`
Rezultat: `C 4`

Explicație

Răspunsul la întrebarea "*Proprietatea 1 a vacii care este trimisă la culcare este* `'X'` *sau* `'Z'`*?*" este *nu* (0), aceasta înseamnă că vaca nu poate avea numerele de ordine 1 și 2, deci trebuie să fie 3 sau 4.

Răspunsul la întrebarea "*Proprietatea 2 a vacii care este dusă la culcare este* `'Y'`*?*" este *da* (1), aceasta înseamnă că vaca poate avea numai numărul de ordine 4, deoarece proprietate 2 a vacii cu numărul de ordine 3 nu este `'Y'`.

După ce este dat răspunsul la ultima întrebare, se poate stabili că rezultatul corect este 4 și programul se oprește după scrierea acestuia la ieșirea standard.

Modalitate de acordare a punctajului

Din punctajul maxim care putea fi obținut pentru un test s-a acordat 30% pentru corectitudinea fișierului de ieșire. Programul a primit tot punctajul pentru corectitudine numai dacă vaca afișată a fost singura vacă ce corespundea răspunsurilor primite. Un program care punea mai mult de 100 de întrebări într-un test nu a primit puncte pe acel test.

Restul de puncte au fost determinate de numărul de întrebări necesare pentru determinarea corectă a vacii. Modul de acordare a punctelor i-a recompensat pe cei care au minimizat numărul de întrebări în cazul cel mai defavorabil.

Timp de execuție: 1 secundă/test

P060306: Roboții uimitori

Sunteți posesorul a doi roboți care se află în labirinturi dreptunghiulare separate. Coordonatele (1, 1) indică pătratul din colțul stânga-sus sau, cu alte cuvinte, din colțul nord-vest al labirintului. În labirintul i ($i \in \{1, 2\}$) există G_i gardieni, care se străduie să prindă robotul respectiv. Fiecare gardian se deplasează continuu mai întâi într-o direcție, iar apoi în cea opusă, de-a lungul unui segment de dreaptă, care reprezintă un traseu prestabilit de patrulare. Scopul vostru este de a determina o secvență de comenzi după execuția cărora roboții vor fi scoși din labirinturile respective fără ca ei să fie capturați de gardieni.

În fiecare minut voi transmiteți ambilor roboți aceeași comandă. Fiecare comandă reprezintă o direcție de deplasare (*Nord*, *Sud*, *Est*, *Vest*). Execuția unei comenzi presupune deplasarea robotului cu un pătrățel în direcția indicată. Dacă în direcția indicată se află un perete, în minutul respectiv robotul rămâne nemișcat. Se consideră că robotul a ieșit din labirint atunci când el se află în exteriorul lui. După ieșirea din labirint, robotul ignoră toate comenzile ulterioare.

În fiecare minut, concomitent cu roboții, gardienii se mută cu un pătrat. Fiecare gardian pornește din pătratul inițial, se deplasează continuu în direcția indicată cu viteza de un pătrat pe minut până când lungimea drumului parcurs este egală cu cea a traseului de patrulare minus unu. În continuare, gardianul se întoarce instantaneu și se deplasează în direcția opusă, către locația de start, unde se întoarce din nou și iarăși repetă traseul de patrulare ș.a.m.d. până când fiecare robot va fi scos din labirintul său. Patrularea nu presupune trecerea gărzii prin pereți sau ieșirea ei din labirint. Deși traseele de patrulare se pot suprapune,

gardienii nu se ciocnesc niciodată, adică, în orice minut, nici o pereche de gardieni nu se va afla în unul și același pătrat sau nu vor face un schimb de pătrate adiacente. Inițial, orice gardian din fiecare labirint se află într-un pătrat care nu coincide cu cel în care se află robotul.

Gardianul capturează un robot atunci când, în minutul curent, ambii se află în același pătrat sau când ei schimbă pătratele adiacente.

Cunoscând cele două labirinturi (fiecare din ele nu mai mare de 20×20), pătratele inițiale ale fiecărui robot și traseele de patrulare ale gardienilor, calculați o secvență de comenzi după execuția căreia roboții vor fi scoși din labirinturi fără ca ei să fie capturați de gardieni.

Minimizați timpul necesar pentru ca ultimul dintre roboți să fie scos din labirint. Dacă roboții ies din labirinturile respective în momente de timp diferite, momentul ieșirii primului dintre ei nu contează.

Date de intrare

Numele fișierului de intrare este **`robots.in`**. Primul set de linii descrie primul labirint, robotul și gărzile din labirint. Într-un mod similar, setul al doilea de linii descrie labirintul al doilea, robotul și gărzile din el.

Prima linie din fișierul de intrare conține două numere întregi R_1 și C_1, separate prin spațiu, care reprezintă numărul de rânduri și numărul de coloane ale primului labirint.

Fiecare dintre următoarele R_1 linii conține câte C_1 caractere care descriu configurația labirintului. Poziția inițială a robotului este redată prin simbolul `'X'`. Simbolul `'.'` reprezintă un pătrat liber, iar simbolul `'#'` reprezintă un perete. În fiecare labirint se află câte un singur robot.

După descrierea labirintului urmează o linie care conține numărul întreg G_1, care indică numărul de gărzi din primul labirint.

Fiecare dintre următoarele G_1 linii descrie un traseu de patrulare, folosind în acest scop trei numere întregi și un caracter, separate printr-un singur spațiu. Primele două numere întregi indică rândul și coloana pătratului de start al gărzii. Numărul al treilea indică numărul de pătrățele (2..4) din traseul de patrulare. Caracterul indică direcția inițială în care se va mișca gardianul: `'N'`, `'S'`, `'E'`, `'W'` (*Nord*, *Sud*, *Est*, *Vest*).

Date de ieșire

Prima linie din fișierul de ieșire **`robots.out`** va conține un singur număr întreg K, reprezentând numărul de comenzi după execuția cărora roboții vor fi scoși din labirint fără ca ei să fie capturați.

Dacă o astfel de secvență există, cea mai scurtă va avea nu mai mult de 10000 de comenzi.

Următoarele K linii vor descrie secvența de comenzi, fiecare linie conținând câte un caracter din mulțimea {`'N'`, `'S'`, `'E'`, `'W'`}.

Dacă o astfel de secvență de comenzi nu există, scrieți la ieșire o singură linie care conține `"-1"`.

Ambii roboți ar trebui să iasă din labirinturi după execuția ultimei comenzi. Ultima comandă ar trebui să scoată din labirint cel puțin unul dintre roboți. Dacă există mai multe secvențe de comenzi care scot roboții din labirinturi într-un timp minim, va fi acceptată oricare din ele.

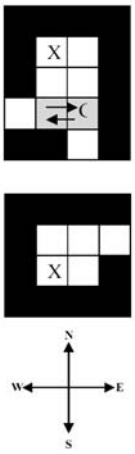
Restricții și precizări

- $1 \le G_1, G_2 \le 10$;
- $1 \le K \le 10.000$.

Exemplu

`robots.in`

```
5 4
####
#X.#
#..#
...#
##.#
1
4 3 2 W
####
#...
#X.#
####
0
```



`robots.out`

```
8
E
N
E
S
S
S
E
S
```

Modalitate de acordare a punctajului

Pentru testele în care secvența de comenzi nu exista, nu s-a acordat punctaj parțial. În celelalte cazuri, punctajul parțial s-a acordat după cum urmează.

Din punctajul maxim care putea fi obținut pentru un test s-a acordat 20% pentru corectitudinea fișierului de ieșire.

Fișierul de ieșire pentru un test s-a considerat corect, dacă acesta a fost formatat corect, conținea cel mult 10000 de comenzi, secvența respectivă de comenzi scotea roboții din labirinturi, iar ultima comandă din secvență scotea cel puțin un robot.

Restul de puncte s-au acordat pentru minimalitatea fișierului de ieșire. Fișierul de ieșire pentru un test s-a considerat a fi minimal, dacă acesta a fost corect și nu exista o altă secvență corectă de comenzi care să fi fost mai scurtă.

Nu s-au acordat puncte pentru minimalitate în cazul secvențelor care nu erau cele mai scurte.

Timp de execuție: 2 secunde/test