PROBLEME propuse pentru REZOLVARE

În perioada 17-24 aprilie a avut loc, la Bacău, Olimpiada Națională de Informatică. În cadrul acestui număr vă prezentăm enunțurile problemelor propuse spre rezolvare în cele patru zile de concurs.

Clasa a IX-a

P050101: Ferma

prof. Maria Niță și prof. Adrian Niță, Oradea

Un fermier are un teren care are forma unui tablou dreptunghiular lung de *M* unități și lat de *N* unități. Pe teren sunt plantați din loc în loc copaci, pe care fermierul nu dorește să-i taie. Dorind să-și supravegheze cultura, fermierul realizează un mic robot de formă pătrată având latura de trei unități pe care îl poate teleghida prin fermă, parcurgând unitate cu unitate o anumită suprafață.

Robotul se poate mișca pe verticală și pe orizontală dar, nu poate trece peste copaci, nu îi poate distruge, nu se poate roti și are nevoie pentru mișcare de o suprafață corespunzătoare dimensiunii lui.

Cerință

Ajutați-l pe fermier să determine suprafața maximă pe care o poate urmări, folosind acest sistem.

Date de intrare

Fișier de intrare: **FERMA.IN**

Linia 1: `N M`

- două numere naturale nenule, separate printr-un spațiu, reprezentând numărul de linii (*N*), respectiv numărul de coloane (*M*);

Liniile 2...*N*+1: $C_1C_2 \ldots C_M$

- aceste *N* linii codifică ferma; fiecare linie conține câte *M* caractere (fără să fie separate prin spații) cu semnificația:
  - `'.'` - teren liber;
  - `'+'` - locul în care este plantat un copac;
  - `'R'` - centrul robotului.

Date de ieșire

Fișier de ieșire: **FERMA.OUT**

Liniile 1...*N*: $C_1C_2 \ldots C_M$

- aceste *N* linii codifică modul în care fermierul poate să-și utilizeze robotul pe terenul său; fiecare linie conține câte *M* caractere (fără să fie separate prin spații) având semnificația:
  - `'.'` - teren neacoperit de robot;
  - `'*'` - teren ce poate fi verificat de robot;
  - `'+'` - loc în care a rămas copacul.

Restricție

- `1 ≤ N, M ≤ 50`

Exemplu

FERMA.IN

```
12 11
...........
...+.....+.
...........
...........
.+.........
...+.......
.+...R.....
.........+.
..+.......+
......+....
...........
......+....
```

FERMA.OUT

```
....*****..
...+*****+.
..*********
..*********
.+*********
...+*******
.+.********
...******+.
..+******.+
******+....
******.....
******+....
```

Timp de execuție: 3 secunde/test

P050102: Fracții

prof. Ovidiu Domșa, Alba Iulia

O proprietate interesantă a fracțiilor ireductibile este că orice fracție se poate obține după următoarele reguli:

- pe primul nivel se află fracția `1/1`;
- pe al doilea nivel, în stânga fracției `1/1` de pe primul nivel, plasăm fracția `1/2` iar în dreapta ei fracția `2/1`;

nivelul 1: 1/1
nivelul 2: 1/2 2/1
- pe fiecare nivel *k* se plasează, sub fiecare fracție *i/j* de pe nivelul anterior, fracția *i* / (*i* + *j*) în stânga și fracția (*i* + *j*) / *j* în dreapta.

nivelul 1: 1/1
nivelul 2: 1/2 2/1
nivelul 3: 1/3 3/2 2/3 3/1

Cerință
Dându-se o fracție oarecare prin numărătorul și numitorul său, determinați numărul nivelului pe care se află fracția sau o fracție echivalentă (având aceeași valoare) cu aceasta.

Date de intrare
Fișier de intrare: **FRACTII.IN**
Linia 1: `N M`
- două numere naturale nenule, separate printr-un spațiu, reprezentând numărătorul și numitorul unei fracții (numărător respectiv numitor).

Date de ieșire
Fișier de ieșire: **FRACTII.OUT**
Linia 1: `niv`
- număr natural nenul, reprezentând numărul nivelului care corespunde fracției.

Restricție
- $1 < N,M \leq 2000000000$ (două miliarde)

Exemple

FRACTII.IN	FRACTII.OUT
13 8	6

FRACTII.IN	FRACTII.OUT
12 8	3

Timp de execuție: 1 secundă/test

P050103: Tablou

prof. Rodica Pintea, București

Generați un tablou bidimensional cu proprietățile:
- conține *N* linii și *N* coloane;
- elementele sale sunt numere naturale nenule;
- suma elementelor este egală cu numărul natural nenul *S*;
- pe nici o linie și pe nici o coloană nu există două elemente identice;
- diferența dintre cel mai mare și cel mai mic element ale tabloului este minimă.

Date de intrare
Fișier de intrare: **TABLOU.IN**
Linia 1: `N S`
- două numere naturale nenule, separate printr-un spațiu, reprezentând numărul de linii și de coloane ale tabloului, respectiv valoarea sumei tuturor elementelor din tablou.

Date de ieșire
Fișier de ieșire: **TABLOU.OUT**
Liniile 1...*N*: nr_{11} nr_{12} … nr_{1N}
nr_{21} nr_{22} … nr_{2N}
...
nr_{N1} nr_{N2} … nr_{NN}
- pe aceste *N* linii se vor scrie elementele tabloului, câte o linie din tablou pe o linie din fișier; elementele se vor separa prin câte un spațiu.

Restricții
- $1 < N \leq 100$
- $0 < S \leq 2^{31}$
- Dacă problema nu are soluție, în fișierul de ieșire se va scrie cifra 0.
- Dacă problema are mai multe soluții, în fișier se va scrie una singură.

Exemplu

TABLOU.IN	TABLOU.OUT
3 51	4 6 7 7 4 5 5 7 6

Timp de execuție: 1 secundă/test

P050104: Competiție dificilă

Angel Proorocu, București

La o competiție au participat *N* concurenți. Fiecare dintre ei a primit un număr de concurs astfel încât să nu existe concurenți cu același număr. Numerele de concurs aparțin mulțimii {1, 2, …, *N*}. Din păcate, clasamentul final a fost pierdut, iar comisia își poate aduce aminte doar câteva relații între unii participanți (de genul "participantul cu numărul 3 a ieșit înaintea celui cu numărul 5").

Cerință
Șeful comisiei are nevoie de un clasament final și vă cere să-l ajutați determinând primul clasament în ordine lexicografică ce respectă relațiile pe care și le amintește comisia.

Date de intrare
Fișier de intrare: **COMPET.IN**
Linia 1: `N M`
- două numere naturale nenule, reprezentând numărul concurenților, respectiv numărul relațiilor pe care și le amintește comisia;

Liniile 2...*M*+1: `i j`
- pe fiecare din aceste *M* linii se află câte două numere naturale nenule *i* și *j*, având semnificația: concurentul cu numărul de concurs *i* a fost în clasament înaintea concurentului cu numărul de concurs *j*.

Date de ieșire
Fișier de ieșire: **COMPET.OUT**

Linia 1: nr_1 nr_2 ... nr_N
- pe această linie se va scrie clasamentul sub forma unui șir de numere naturale nenule, separate prin câte un spațiu, reprezentând numerele de concurs ale concurenților, în ordine, de la primul clasat la ultimul.

Restricții
- $1 < N \leq 1000$
- se garantează corectitudinea datelor de intrare și faptul că există totdeauna o soluție.

Exemple

COMPET.IN	COMPET.OUT
3 1 1 2	1 2 3

COMPET.IN	COMPET.OUT
4 2 2 1 3 4	2 1 3 4

Timp de execuție: 1 secundă/test

P050105: Cuvinte

prof. Maria Niță și prof. Adrian Niță, Oradea

Se consideră o listă având un număr cunoscut de cuvinte. Din această listă s-au ales două cuvinte oarecare. Se dorește transformarea primului cuvânt în cel de-al doilea, trecând prin cuvinte intermediare, existente în lista dată. Pentru transformarea unui cuvânt în altul se pot folosi următoarele operații: schimbarea, adăugarea sau ștergerea unui singur caracter.

Cerință
Dându-se o listă de cuvinte și două cuvinte din aceasta, găsiți cea mai scurtă secvență de operații care transformă primul cuvânt în cel de-al doilea folosind operațiile permise.

Date de intrare
Fișierul de intrare: **CUVINTE.IN**

Linia 1: N P Q
- trei numere naturale nenule, reprezentând numărul cuvintelor din listă (N), poziția primului cuvânt în listă (P), respectiv poziția celui de-al doilea cuvânt în listă (Q);

Liniile 2...N+1: cuvant
- aceste N linii conțin fiecare câte un cuvânt, aparținând listei.

Date de ieșire
Fișier de ieșire: **CUVINTE.OUT**

Linia 1: M
- număr natural, reprezentând numărul minim de pași necesari pentru a ajunge de la primul cuvânt la al doilea;

Liniile 2...M+2: cuv_i
- pe aceste linii apar în ordine cuvintele dintr-o secvență ce respectă cerința problemei (câte un cuvânt pe linie).

Restricții
- $2 \leq N \leq 100$
- $1 \leq M,P,Q \leq 100$
- numărul maxim de caractere dintr-un cuvânt este 100;
- dacă nu există soluție, în fișierul de ieșire se va scrie numărul 0 (zero);
- dacă sunt mai multe secvențe de lungime minimă, în fișier se va scrie una singură.

Exemple

CUVINTE.IN	CUVINTE.OUT
7 1 5 car cer cerc mar mare rosu inrosit	2 car mar mare

CUVINTE.IN	CUVINTE.OUT
7 1 6 car cer cerc mar mare	0

Timp de execuție: 2 secunde/test

P050106: Grup

prof. Emanuela Cerchez și prof. Marinel Șerban, Iași

Administratorul rețelei cu N calculatoare de la SRI împarte, din motive strategice, aceste calculatoare în mai multe grupuri. De fapt, important este doar numărul de grupuri și numărul de calculatoare din fiecare grup, așa că împărțirea este descrisă prin șirul numerelor de calculatoare din fiecare grup, ordonat crescător. Periodic, el procedează la o nouă împărțire pe grupe a calculatoarelor. Dintre toate împărțirile posibile ale calculatoarelor în grupuri putem alege ca următoare împărțire doar aceea a cărei descriere precede sau succede lexicografic imediat împărțirii curente.

Notă: Spunem că șirul $x_1x_2...x_p$ precede lexicografic șirul $y_1y_2...y_k$ dacă:

a) există un indice j, astfel încât $x_i = y_i$ pentru $i < j$ și $x_j < y_j$

sau

b) $p < k$ și $x_i = y_i$ pentru toți indicii $i \leq p$

Exemple:

a) 3 7 2 5 precede lexicografic 3 7 4 1 6 2

b) 1 2 3 precede lexicografic 1 2 3 4

Cerință
Dându-se o împărțire în grupe a celor N calculatoare, determinați cele două variante candidate pentru împărțirea următoare.

Date de intrare
Fișier de intrare: **GRUP.IN**
Linia 1: `N k`
- numere naturale nenule, reprezentând numărul total (N) al calculatoarelor din rețea și numărul (k) de grupe.

Linia 2: $g_1\ g_2\ \ldots\ g_k$
- numărul calculatoarelor din fiecare grupă.

Date de ieșire
Fișier de ieșire: **GRUP.OUT**
Linia 1: `p`
- numărul de grupe din împărțirea care precede lexicografic imediat împărțirea dată;

Linia 2: $h_1\ h_2\ \ldots\ h_p$
- numărul de calculatoare din cele p grupe ale împărțirii precedente;

Linia 3: `u`
- numărul de grupe din împărțirea care succede lexicografic imediat împărțirea dată;

Linia 4: $t_1\ t_2\ \ldots\ t_u$
- numărul de calculatoare din cele `u` grupe ale împărțirii următoare.

Restricții
- $2 \le N \le 1000$
- $g_1+g_2+\ldots+g_k=h_1+h_2+\ldots+h_p=t_1+t_2+\ldots+t_u=N$
- $1 \le g_1 \le g_2 \le \ldots \le g_k$; $1 \le h_1 \le h_2 \le \ldots \le h_p$; $1 \le t_1 \le t_2 \le \ldots \le t_u$
- $1 < k < N$
- $1 \le p, u \le N$

Exemplu

GRUP.IN	GRUP.OUT
14 3	3
2 6 6 `	2 5 7
	2
	2 12

Timp de execuție: 1 secundă/test

Clasa a X-a

P050107: Alpinistul

Bogdan Stroe, București

Un alpinist ar vrea să cucerească un vârf cât mai înalt din Munții Carpați. Din păcate îi este frică să urce de pe o stâncă pe alta alăturată, dacă diferența de altitudine depășește `100` de metri. În schimb el coboară fără frică de pe o stâncă pe alta alăturată, indiferent de diferența de altitudine.

Alpinistul are harta muntelui pe care vrea să-l escaladeze. Harta este codificată sub forma unui caroiaj în care în pătrățele sunt notate altitudinile punctelor (înălțimile stâncilor) de pe hartă, date în metri față de nivelul mării. Alpinistul poate porni din orice pătrățel de pe hartă cu altitudine `0` (aflat la nivelul mării) și poate efectua un pas doar într-o porțiune de teren corespunzătoare unui pătrățel de pe hartă, învecinat pe verticală sau pe orizontală cu pătrățelul în care se află, cu condiția să nu îi fie frică.

Cerință
Alpinistul apelează la ajutorul vostru pentru a afla traseul de lungime minimă pe care trebuie să-l urmeze pentru a escalada un vârf cât mai înalt.

Date de intrare
Fișier de intrare: **ALPINIST.IN**
Linia 1: `M N`
- două numere naturale nenule, separate printr-un spațiu, reprezentând dimensiunile caroiajului corespunzător hărții;

Liniile 2...M+1: $v_1\ v_2\ \ldots\ v_N$
- pe aceste M linii sunt scrise câte N numere naturale, separate prin câte un spațiu, reprezentând valorile asociate caroiajului care codifică harta (linie după linie).

Date de ieșire
Fișier de ieșire: **ALPINIST.OUT**
Linia 1: `I`
- număr natural ce reprezintă înălțimea maximă la care poate ajunge alpinistul;

Linia 2: $X_P\ Y_P\ X_D\ Y_D$
- patru numere naturale nenule, separate prin câte un spațiu, reprezentând coordonatele pătrățelului din care pleacă alpinistul și coordonatele pătrățelului având înălțimea maximă în care poate ajunge; prin coordonatele pătrățelului se înțeleg numărul liniei și cel al coloanei pe care se află pătrățelul în caroiaj;

Linia 3: `L`
- număr natural reprezentând lungimea drumului; lungimea unui drum se definește ca fiind egal cu numărul pătrățelelor străbătute de alpinist;

Linia 4: $D_1\ D_2\ \ldots\ D_L$
- L caractere, separate prin câte un spațiu, reprezentând direcția în care se mișcă alpinistul la fiecare pas i, $i = 1, 2, \ldots, L$; caracterele pot fi: `'N'`(corespunzător unei mișcări în sus) `'S'` (corespunzător unei mișcări în jos), `'W'` (corespunzător unei mișcări la stânga) și `'E'` (corespunzător unei mișcări la dreapta).

Restricții
- $2 \le M, N \le 100$
- $0 \le v_i \le 1000$, $i = 1, 2, \ldots, N$ (pe fiecare din cele M linii)
- dacă sunt mai multe drumuri de lungime minimă, în fișier se va scrie unul singur.

Exemplu

ALPINIST.IN	ALPINIST.OUT
3 5	250
0 101 2 14 100	1 1 2 4
50 149 149 250 200	8
0 100 10 400 10	S E E N E E S W

Timp de execuție: 1 secundă/test

P050108: Asfaltare

prof. Roxana Tîmplaru, Craiova

Un primar proaspăt ales vrea să dovedească electoratului său că votându-l, cetățenii au făcut o alegere bună. În acest scop el a decis să reasfalteze străzile dintre N edificii importante din oraș, numerotate de la 1 la N. Între oricare două dintre aceste edificii există o singură stradă cu două sensuri de circulație. Edificiul numerotat cu 1 este primăria.

Primarul cere consilierilor să stabilească toate traseele pe care reasfaltarea străzilor o poate urma printre cele N edificii, știind că are H străzi "preferate" pe care trebuie să le asfalteze în mod obligatoriu. Se știe că oricare două străzi preferate nu au capete comune. Traseele care se vor reasfalta trebuie să pornească de la primărie, să ajungă o singură dată la fiecare din celelalte N-1 edificii și să se întoarcă tot la primărie.

Cerință

Determinați numărul traseelor distincte, respectând cerințele de mai sus.

Date de intrare

Fișier de intrare: **ASFALT.IN**

Linia 1: `N H`

- două numere naturale, separate printr-un spațiu, reprezentând numărul edificiilor (N), respectiv numărul străzilor preferate ale primarului (H).

Date de ieșire

Fișier de ieșire: **ASFALT.OUT**

Linia 1: `x`

- număr întreg pozitiv, reprezentând numărul traseelor distincte, posibile.

Restricții

- $3 \le N \le 1000$
- $0 \le H \le N/2$
- dacă un traseu este diferit de un altul doar prin direcția în care se parcurge drumul, pornind de la primărie și revenind aici, acesta se consideră identic cu primul. De exemplu, traseul `1-2-3-4-1` este identic cu traseul `1-4-3-2-1`.

Exemplu

ASFALT.IN	ASFALT.OUT
4 1	2

Timp de execuție: 5 secunde/test.

P050109: Oracolul decide!

prof. Doru Popescu Anastasiu, Slatina

La un concurs participă N concurenți. Fiecare concurent primește o foaie de hârtie pe care va scrie un cuvânt având cel mult 100 de caractere (litere mici ale alfabetului englez). Cuvintele vor fi distincte.

Pentru departajare, concurenții apelează la un oracol. Acesta produce și el un cuvânt. Va câștiga concurentul care a scris cuvântul "cel mai apropiat" de cel al oracolului.

Gradul de "apropiere" dintre două cuvinte este lungimea subcuvântului comun de lungime maximă. Prin subcuvânt al unui cuvânt dat se înțelege un cuvânt care se poate obține din cuvântul dat, eliminând 0 sau mai multe litere și păstrând ordinea literelor rămase.

Cerință

Se cunosc cuvântul c_0 produs de oracol și cuvintele c_i, i = 1, …, N scrise de concurenți. Pentru a ajuta comisia să desemneze câștigătorul, se cere ca pentru fiecare valoare i să identificați pozițiile literelor ce trebuie șterse din c_0 și din c_i astfel încât prin ștergere să se obțină unul dintre subcuvintele comune de lungime maximă.

Date de intrare

Fișier de intrare: **ORACOL.IN**

Linia 1: `N`

- număr natural nenul, reprezentând numărul concurenților;

Linia 2: c_0

- cuvântul produs de oracol;

Liniile 3...*N*+2: `cuvant`

- pe aceste N linii se află cuvintele scrise de cei N concurenți, un cuvânt pe o linie.

Date de ieșire

Fișier de ieșire: **ORACOL.OUT**

Liniile 1...2**N*: *pozițiile literelor ce trebuie șterse*

- pe fiecare linie i (i = 1, 3, ..., 2*N - 1) se vor scrie numere naturale nenule, separate prin câte un spațiu, reprezentând pozițiile de pe care se vor șterge litere din cuvântul produs de oracol; pe fiecare linie j (j = 2, 4, ..., 2*N) se vor scrie numere naturale nenule, separate prin câte un spațiu, reprezentând pozițiile de pe care se vor șterge litere din cuvântul concurentului cu numărul j/2.

Restricții

- $2 \le N \le 100$
- dacă există mai multe soluții, în fișier se va scrie una singură.
- dacă dintr-un cuvânt nu se va tăia nici o literă, linia respectivă din fișierul de intrare va rămâne vidă.

Exemplu

ORACOL.IN	ORACOL.OUT
3	3
abc	3 4
abxd	
aabxyc	1 4 5
acb	3
	2

Timp de execuție: 1 secundă/test

P050110: Alipiri

prof. Gelu Manolache, Piatra Neamț

Spunem că numărul natural nenul n_1 se poate alipi pe K biți cu numărul natural nenul n_2 dacă reprezentările lor binare au fiecare cel puțin K biți (primul bit fiind 1), iar ultimii K biți ai numărului n_1 coincid cu primii K biți ai numărului n_2. După alipire rezultă un alt număr, format din concatenarea lui n_1 cu n_2, dar în care secvența comună de K biți apare o singură dată și în ea fiecare bit este înlocuit cu complementarul său (0 devine 1 și invers).

Exemplu: Numărul $78=1001110_2$ se poate alipi pe trei biți cu $25=11001_2$ și rezultă numărul $100100101_2=293$.

Numerele $nr_1, nr_2, ..., nr_M$ formează o listă de alipiri pe K biți, de lungimea $M \geq 1$, cu finalul P, dacă nr_1 se alipește pe K biți cu nr_2, rezultatul se alipește pe K biți cu nr_3, ..., rezultatul se alipește pe K biți cu nr_M, iar rezultatul final este numărul P.

Notă: Dacă $M = 1$, atunci trebuie să avem $nr_1 = P$.

Cerințe

Pentru un șir dat de numere naturale nenule și valorile naturale nenule K și P, se cere:

a) să se determine lungimea maximă a unei liste de alipiri pe K biți, formată cu elemente din șirul dat, nu neapărat în ordinea din acest șir;

b) să se verifice dacă se poate obține, cu numere din șirul dat, o listă de alipiri pe K biți cu finalul P; în caz afirmativ să se precizeze o listă de lungime minimă de alipiri pe K biți cu finalul P.

Date de intrare

Fișier de intrare: **ALIPIRI.IN**

Linia 1: `N K P`

- trei numere naturale nenule, separate prin câte un spațiu, reprezentând numărul N al numerelor date, lungimea K a secvențelor comune din alipiri, respectiv valoarea P a numărului care trebuie obținut prin alipiri;

Linia 2: $nr_1\ nr_2\ \ldots\ nr_N$

- N numere naturale nenule, separate prin câte un spațiu, reprezentând șirul dat.

Date de ieșire

Fișier de ieșire: **ALIPIRI.OUT**

Linia 1: `M`

- număr natural nenul, reprezentând rezultatul de la cerința a);

Linia 2: *mesaj*

- dacă cerința b) poate fi satisfăcută, pe această linie se va scrie `DA`, în caz contrar se va scrie `NU`;

Linia 3: $nr_1\ nr_2\ \ldots$

- dacă pe linia 2 s-a scris `DA`, pe această linie se va scrie o listă de alipiri (pe K biți) de lungime minimă cu finalul P; numerele din listă se vor separa prin câte un spațiu.

Restricții

- $1 \leq K \leq 8, 1 \leq N \leq 9, 1 \leq P \leq 2000000000$
- dacă cerința b) este satisfăcută de mai multe liste (de lungime minimă), în fișier se va scrie una singură;
- pentru ambele cerințe fiecare număr din șirul de intrare poate apărea cel mult o dată în listele de alipiri considerate.

Exemplu

ALIPIRI.IN	ALIPIRI.OUT
4 3 291	4
14 59 36 31	DA
	59 31 14

Timp de execuție: 10 secunde/test

P050111: Drum

prof. Doru Popescu Anastasiu, Slatina

În regatul *XLand* orașele erau înconjurate de ziduri în formă de poligoane convexe. Împăratul a dispus construirea unui drum de legătură directă între capitală și un alt oraș dat. Fiecare extremitate a drumului poate fi orice punct situat pe zidul orașului, respectiv capitalei. Lungimea drumului este distanța dintre extremitățile sale.

Cerință

Determinați cel mai scurt drum dintre capitală și orașul dat.

Date de intrare

Fișier de intrare: **DRUM.IN**

Linia 1: `K1`

- număr natural nenul, reprezentând numărul de colțuri ale zidurilor capitalei;

Linia 2: $x_1\ y_1\ x_2\ y_2\ \ldots\ x_{K1}\ y_{K1}$

- $K1$ perechi de numere întregi, separate prin câte un spațiu, reprezentând coordonatele vârfurilor zidurilor capitalei;

Linia 3: `K2`

- număr natural nenul, reprezentând numărul de colțuri ale zidurilor orașului dat;

Linia 4: $x_1\ y_1\ x_2\ y_2\ \ldots\ x_{K2}\ y_{K2}$

- $K2$ perechi de numere întregi, separate prin câte un spațiu, reprezentând coordonatele vârfurilor zidurilor acestui oraș.

Date de ieșire

Fișier de ieșire: `DRUM.OUT`

Linia 1: $x_1\ y_1\ x_2\ y_2$

- patru numere reale trunchiate la patru zecimale, separate prin câte un spațiu, reprezentând extremitățile drumului de legătură respectiv.

Restricții

- $2 \leq K1, K2 \leq 20$
- coordonatele vârfurilor zidurilor ce înconjoară orașul, respectiv capitala sunt numere întregi aparținând intervalului [-100, 100] și sunt date fie în sensul deplasării

acelor de ceasornic, fie în sens invers deplasării acelor de ceasornic;
- capitala și orașul nu au nici un punct comun (nu au puncte interioare comune și nu au puncte comune pe zidurile lor).

Exemplu

DRUM.IN	DRUM.OUT
4 3 4 3 2 5 2 5 4 4 8 3 8 6 11 6 11 3	5.0000 3.5000 8.0000 3.5000

Timp de execuție: 1 secundă/test

P050112: Pavări

prof. Doru Popescu Anastasiu, Slatina

Se dă un dreptunghi cu lungimea egală cu 2*N* centimetri și lățimea egală cu trei centimetri.

Cerință

Să se determine numărul *M* al pavărilor distincte cu dale dreptunghiulare care au lungimea egală cu un centimetru și lățimea egală cu doi centimetri.

Date de intrare

Fișier de intrare: **PAVARI.IN**

Linia 1: `N`

- număr natural nenul, reprezentând jumătatea lungimii dreptunghiului.

Date de ieșire

Fișier de ieșire: **PAVARI.OUT**

Linia 1: `M`

- număr natural nenul, reprezentând numărul modalităților de a pava dreptunghiul.

Restricții

- $1 \le N \le 100$

Exemplu

PAVARI.IN	PAVARI.OUT
2	11

Timp de execuție: 1 secundă/test

Clasele a XI-a și a XII-a

P050113: Comparări

Bogdan Dumitru, București

Fie un șir de *N* numere întregi. Numim al doilea maxim al șirului cel mai mare număr din șir, cu excepția maximului. Pentru a afla maximul și al doilea maxim, veți efectua comparări între elementele șirului.

Având la dispoziție numai $N + \lceil \log_2 N \rceil - 2$ comparări, aflați pozițiile pe care le ocupă maximul și al doilea maxim. Prin $\lceil \log_2 N \rceil$ s-a notat *partea întreagă superioară a lui "logaritm în baza 2 din N"*.

Cerință

Aflați cele două numere, efectuând cel mult $N + \lceil \log_2 N \rceil - 2$ comparări. Pentru a putea efectua comparările, comisia vă pune la dispoziție funcția `int Compara(int i,int j)` (pentru cei care programează în *C* sau *C++*), respectiv funcția `Compara(i,j:Integer):Integer;` (pentru programatorii în *Pascal*), care compară numărul de pe poziția *i* cu cel de pe poziția *j* și returnează `1`, dacă elementul aflat pe poziția *i* este mai mare sau egal cu cel de pe poziția *j*, respectiv `-1`, în caz contrar.

Programul trebuie să apeleze la început funcția `GetN` care returnează numărul de elemente din șir. Apelul va fi de forma `N=GetN()`, pentru programatorii în *C/C++*, respectiv `N:=GetN`, pentru programatorii în *Pascal*, unde am presupus că *N* este variabilă de tip **int**, respectiv `Integer`, în care este memorată lungimea șirului. Este esențial ca apelul funcției `GetN` să preceadă apelurile funcției `Compara` și, în plus, să existe un singur apel al funcției `GetN` .

Programatorii în *C/C++* vor găsi fișierul **COMPAR.H**, iar cei care lucrează în *Pascal* vor găsi fișierul **COMPAR.PAS** reprezentând sursele unit-ului comisiei.

În fișierul **COMPAR.OUT** veți scrie pozițiile pe care se află maximul șirului și al doilea maxim, separate printr-un spațiu.

Pentru a folosi unit-ul, trebuie să creați fișierul **COMPAR.IN**, în care veți scrie pe prima linie numărul *N*, iar pe a doua linie elementele șirului, separate prin câte un spațiu.

Versiunea *Pascal* a unit-ului este:

```
Unit compar;

interface

const my_nmax=1000;
var my_a:array[1..my_nmax+20] of Integer;
    my_apelat,my_n,my_napel,my_apel:Integer;
function GetN:Integer;
function compara(i,j:Integer):Integer;

implementation

function GetN:Integer;
var f:Text;
    i,log,tmp:Integer;
begin
  my_apelat:=1;
  Assign(f,'COMPAR.IN');
  Reset(f);
  Readln(f,my_n);
  for i:=1 to my_n do
    Read(f,my_a[i]);
```

```
  Close(f);
  my_apel:=0;
  tmp:=1;
  log:=0;
  while tmp<my_n do
  begin
    tmp:=tmp*2;
    log:=log+1
  end;
  my_napel:=my_n+log-2;
  getn:=my_n
end;

function compara(i,j:Integer):Integer;
var f:Text;
begin
  if my_apelat=0 then
  begin
    Writeln('Nu ati apelat functia GetN!');
    Halt
  end;
  my_apel:=my_apel+1;
  if my_apel>my_napel then
  begin
    Writeln('Prea multe apeluri!');
    Halt
  end;
  if (i<1) or (i>my_n) or
     (j<1) or (j>my_n) then
  begin
    Writeln('Apel ilegal!');
    Halt
  end;
  if my_a[i]>=my_a[j] then
    compara:=1
  else compara:=-1
end;

End.
```

Implementarea *C/C++* a aceleași funcții este următoarea:

```
#include <stdio.h>
#include <math.h>
#include <process.h>
#define my_nmax 1000

int my_a[my_nmax+20];
int my_apelat,my_n,my_napel,my_apel;

extern int GetN(void){
  FILE *f;
  int i,log,tmp;
  my_apelat=1;
  f=fopen("COMPAR.IN","r");
  fscanf(f,"%d",&my_n);
  for (i=1;i<=my_n;i++)
    fscanf(f,"%d",&my_a[i]);
  fclose(f);
  my_apel=0;
  tmp=1;
  log=0;
  while (tmp<my_n){
    tmp=tmp*2;
    log++;
  }
  my_napel=my_n+log-2;
  return my_n;
}

extern int Compara(int i,int j){
  FILE *f;
  if (!my_apelat){
    puts("Nu ati apelat functia GetN()!");
    exit(0);
  }
  my_apel=my_apel+1;
  if (my_apel>my_napel){
    puts("Prea multe apeluri!");
    exit(0);
  }
  if (i<1 || i>my_n || j<1 || j>my_n){
    puts("Apel ilegal!");
    exit(0);
   }
  if (my_a[i]>=my_a[j])
    return 1;
  else return -1;
}
```

Restricții

- $2 \le N \le 1000$

Exemple

Dacă șirul considerat este `5 3 4`, atunci o executare corectă a programului este următoarea:

`N = GetN()`, obțineți `N=3`;
`Compara(1,2)`, se returnează `1`;
`Compara(1,3)`, se returnează `1`;
`Compara(2,3)`, se returnează `-1`;

În acest moment ați epuizat numărul de comparări permise (`3+2-2=3` comparări) și în fișierul **`COMPAR.OUT`** veți scrie numerele `1` și `3`, separate printr-un spațiu, reprezentând pozițiile pe care se află maximul șirului (`5`), respectiv al doilea maxim (`4`).

Dacă șirul considerat este (`5`, `7`, `7`), o executare corectă a programului este următoarea:
`N = GetN()`, obțineți `N=3`;
`Compara(1,2)`, se returnează `-1`;

`Compara(2,3)`, se returnează `1`;
`Compara(1,3)`, se returnează `-1`;

În acest moment ați epuizat numărul de comparări permise (`3+2-2=3` comparări) și în fișierul **`COMPAR.OUT`** veți scrie numerele `2` și `3`, separate printr-un spațiu, reprezentând pozițiile pe care se află maximul șirului (`7`), respectiv al doilea maxim (`7`).

Observație

Pentru acest exemplu, mai există o soluție corectă, cea în care considerați maximul (`7`) pe poziția `3` și al doilea maxim (`7`) pe poziția `2`.

Timp de execuție: 1 secundă/test

Se garantează că `1008` apeluri ale funcției `Compara`, împreună cu un apel al funcției `GetN` nu durează mai mult de 0.1 secunde.

P050114: Relee

șef lucrări Stelian Ciurea, Sibiu

Fie date altitudinile a *N* puncte, situate în linie dreaptă de-a lungul axei *Ox*, astfel încât ele corespund unor abscise naturale consecutive. Primul punct are abscisa `1`. Din acest punct trebuie trimisă o rază laser în ultimul punct (cel de abscisă *N*). Raza se propagă doar în linie dreaptă. Pentru a "ocoli" punctele având altitudini care împiedică trecerea razei, în anumite puncte se montează relee care schimbă unghiul sub care se propagă raza, cu scopul ca ea să poată trece de vârfurile care se află în drumul ei.

Releele se vor monta în oricare dintre punctele date, mai puțin în primul punct, de unde raza poate porni sub orice unghi și în ultimul punct unde raza poate fi recepționată sub orice unghi.

S-a observat că dacă în anumite puncte releul se montează pe un pilon, numărul releelor necesare se poate micșora. Toți pilonii care se vor monta au aceeași înălțime dată *H*.

Cerință

Determinați numărul minim de relee pentru ca raza să ajungă din punctul inițial în cel final, precum și punctele în care acestea se vor monta. În cazul în care există mai multe soluții cu același număr minim de relee, se va alege cea cu număr minim de piloni.

Date de intrare

Fișier de intrare: **`RELEE.IN`**

Linia 1: `N H`
- două numere naturale nenule, reprezentând numărul punctelor (`N`), respectiv înălțimea pilonilor (`H`);

Linia 2: $A_1\ A_2\ \ldots\ A_N$
- *N* numere întregi, separate prin câte un spațiu, reprezentând altitudinile (înălțimile) punctelor.

Date de ieșire

Fișier de ieșire: **`RELEE.OUT`**

Linia 1: `NRrelee`
- număr natural, reprezentând numărul releelor care se vor monta, fără să fie înălțate pe piloni;

Linia 2: `NRpiloni`
- număr natural, reprezentând numărul pilonilor care se vor monta;

Linia 3: $C_1\ C_2\ \ldots\ C_{NRrelee}$
- numere naturale nenule, reprezentând numărul de ordine al punctelor unde se vor monta relee, fără să fie înălțate pe piloni;

Linia 4: $D_1\ D_2\ \ldots\ D_{NRpiloni}$
- numere naturale nenule, reprezentând numărul de ordine al punctelor unde releele se vor monta pe piloni.

Dacă valoarea `NRrelee` este `0` atunci linia a treia va fi vidă, iar dacă valoarea `NRpiloni` este `0`, atunci linia a patra va fi vidă.

Restricții
- $1 \le N \le 200$
- $1 \le H \le 500$
- $1 \le A_i \le 2500$ $i = 1, 2, \ldots, N$
- dacă trei vârfuri sunt coliniare, atunci pe cel din mijloc nu trebuie amplasat un releu.

Exemplu

`RELEE.IN`	**`RELEE.OUT`**
`9 2`	`1`
`3 2 6 6 4 3 5 3 2`	`1`
	`7`
	`4`

Timp de execuție: 2 secunde/test

P050115: Telecomanda

prof. Emanuela Cerchez și prof. Marinel Șerban, Iași

Cu ocazia olimpiadei, televiziunea locală organizează un nou joc în direct. Organizatorii utilizează un calculator, care generează și afișează pe un monitor două numere de cel mult 100 de cifre fiecare (*N*1 și *N*2). Fiecare concurent dispune de o telecomandă prevăzută cu un afișaj de o cifră și cu anumite taste. Telecomanda are și o memorie, în care sunt reținute în ordine cifrele obținute de concurenți. Cifrele primului număr (*N*1) sunt afișate succesiv pe afișajul telecomenzii fiecărui concurent, în ordine de la stânga la dreapta. Concurenții trebuie să transforme primul număr, obținând în memoria telecomenzii proprii pe cel de al doilea, utilizând tastele pe care le au la dispoziție pe telecomandă. După efectuarea unei operații asupra cifrei curente (cea de pe afișaj), pe afișaj apare automat următoarea cifră din *N*1 (dacă mai există).Efectele apăsării tastelor sunt:
- + *urmat de o cifră*: se generează suma dintre cifra de pe afișaj și cifra tastată (operație posibilă doar dacă suma este tot o cifră); cifra sumă este reținută în memorie;
- - *urmat de o cifră*: se generează diferența dintre cifra de pe afișaj și cifra tastată (operație posibilă doar dacă se obține tot o cifră); cifra obținută este reținută în memorie;

- * *urmat de o cifră*: se reține în memorie valoarea tastei care se acționează după tasta *; deoarece asupra cifrei curente din *N*1 nu se efectuează nici o operație, aceasta nu dispare de pe afișaj;
- /: șterge cifra curentă din *N*1;
- #: se șterg din *N*1 cifra curentă și toate cifrele care urmează, până la sfârșit.
- =: se copiază în memorie cifra curentă.

Acțiunea se încheie atunci când toate cifrele lui *N*1 au fost prelucrate. Se obține o soluție dacă în memoria telecomenzii se află cifrele numărului *N*2. O soluție este optimă dacă numărul de taste acționate este minim. Câștigătorii jocului sunt acei concurenți care descoperă o soluție optimă.

Cerință
Date fiind *N*1 și *N*2, scrieți un program care să determine o soluție optimă de transformare a numărului *N*1 în numărul *N*2.

Date de intrare
Fișier de intrare: **TELE.IN**
Linia 1: `N1`
- numărul inițial

Linia 2: `N2`

Date de ieșire
Fișier de ieșire **TELE.OUT**
Linia 1: `min`
- număr natural nenul, reprezentând numărul minim de taste acționate pentru transformarea lui *N*1 în *N*2.

Linia 2: $t_1 t_2 \ldots t_{min}$
- succesiune de `min` caractere, care reprezintă tastele acționate; între caractere nu vor apărea separatori.

Exemplu

TELE.IN	TELE.OUT
372	4
78	/=+6

Timp de execuție: 1 secundă/test

P050116: Entries

Dumitru Bogdan, București

Se consideră un graf care inițial este format din *P* noduri izolate, etichetate de la 1 la *P*. Se mai consideră *N* intrări, unde intrare poate însemna:
- *comandă* - o comandă are forma `I+J`, cu semnificația că în graf se adaugă muchia care unește nodurile *I* și *J* (dacă *I* și *J* erau deja unite în acel moment, nu se întreprinde nici o acțiune);
- *întrebare* - o întrebare este de forma `'I?J'`, adică se întreabă dacă în acel moment *I* și *J* sunt în aceeași componentă conexă.

Se pleacă deci de la un graf inițial format din noduri izolate, care pe parcurs se "unifică". Tot pe parcurs sunteți întrebați dacă anumite perechi de noduri sunt sau nu în aceeași componentă conexă.

Din fișierul **ENTRIES.IN** veți citi de pe prima linie numărul *N* de intrări. Pe următoarele *N* linii se găsesc intrările, câte una pe linie. O intrare este codificată prin trei numere separate prin câte un blanc. Primele două numere reprezintă nodurile *I* și *J* (numere întregi, cuprinse între 1 și *P*), iar al treilea este `1` dacă intrarea este o comandă, respectiv `2` dacă intrarea este o întrebare.

La fiecare întrebare, veți scrie pe o linie separată în fișierul **ENTRIES.OUT** numărul `1` dacă nodurile despre care ați fost întrebați sunt în acel moment în aceeași componentă conexă, respectiv numărul `0` în caz contrar.

Restricții
- `1 ≤ N ≤ 5000`
- `1 ≤ P ≤ 10000000`

Exemplu

ENTRIES.IN	ENTRIES.OUT
9	0
1 2 2	0
1 2 1	1
3 7 2	0
2 3 1	1
1 3 2	0
2 4 2	
1 4 1	
3 4 2	
1 7 2	

Timp de execuție: 1 secundă/test

P050117: Robot

prof. Emanuela Cerchez, Iași

Un robot punctiform se poate deplasa, în plan, în linie dreaptă în orice direcție. Robotul se găsește într-o poziție inițială *S* și trebuie să ajungă într-o poziție finală *F*, evitând coliziunile cu obstacolele existente în teren. Obstacolele sunt suprafețe poligonale convexe, cu interioarele și frontierele disjuncte. Spunem că robotul a intrat în coliziune cu un obstacol dacă poziția sa devine interioară obstacolului. Prin urmare, dacă robotul se deplasează de-a lungul unui obstacol, nu intră în coliziune cu acesta.

Cerință
Scrieți un program care să determine cel mai scurt traseu pe care robotul îl poate urma de la poziția sa inițială *S* la poziția sa finală *F*, fără a intra în coliziune cu nici un obstacol. Traseul va fi precizat prin succesiunea punctelor critice (punctul inițial, punctele în care robotul își schimbă direcția și punctul final). Lungimea traseului este egală cu suma lungimilor segmentelor care îl compun.

Date de intrare
Fișier de intrare: **ROBOT.IN**

Linia 1: x_S y_S
• coordonatele poziției inițiale a robotului;
Linia 2: x_F y_F
• coordonatele poziției finale a robotului;
Linia 3: n
• numărul de obstacole;
Linia 4: k_1
• numărul de vârfuri ale primului obstacol;
Liniile 5...4+k_1: x_{i1} y_{i1}
• k_1 linii care conțin câte două numere care reprezintă coordonatele vârfurilor primului obstacol;
Linia 5+k_1: k_2
• numărul de vârfuri ale celui de-al doilea obstacol;
Liniile 6+k_1...5+k_1+k_2: x_{i2} y_{i2}
• k_2 linii care conțin câte două numere care reprezintă coordonatele vârfurilor celui de-al doilea obstacol;
...
Linia 4+n+k_1+...+k_{n-1}: k_n
• numărul de vârfuri ale ultimului obstacol;
Liniile 5+n+k_1+...+k_{n-1}...4+n+k_1+...+k_n: x_{in} y_{in}
• k_n linii care conțin câte două numere care reprezintă coordonatele vârfurilor ultimului obstacol.

Date de ieșire
Fișier de ieșire: **ROBOT.OUT**
Linia 1: nr
• numărul de puncte de pe traseu;
Liniile 2..*nr*+1: x_i y_i
• coordonatele punctelor critice de pe traseu.

Restricții
• n este un număr natural, $0 \le N \le 50$
• $k_1+k_2+...+k_n \le 200$
• x_i, y_i sunt numere reale, $|x_i|, |y_i| \le 100000$
• punctele *S* și *F* nu se află în interiorul nici unui obstacol
• coordonatele se vor scrie în fișierul de ieșire cu trei zecimale semnificative
• dacă există mai multe trasee de lungime minimă, în fișierul de ieșire se va scrie o singură soluție.

Exemplu

ROBOT.IN	ROBOT.OUT
10 5 -10 -10 1 3 0 0 0 10 10.5 0	3 10 5 10.5 0 -10 -10

Timp de execuție: 1 secundă/test

P050118: Text mare

Radu Ștefan, Brașov

Se dă un șir de caractere, ce reprezintă o frază; între cuvinte nu apar separatori. De asemenea, se dă un vocabular având cel mult 32000 de cuvinte; fiecare cuvânt este format din cel mult 16 caractere. Fraza poate avea cel mult 32000 caractere, care sunt litere mici din alfabetul latin. Cuvintele din vocabular nu sunt neapărat diferite și nu apar într-o ordine prestabilită.

Cerință
Despărțiți fraza într-un număr minim de cuvinte; toate aceste cuvinte trebuie să existe în vocabularul dat.

Date de intrare
Fișier de intrare: **TEXTMARE.IN**
Linia 1: fraza
• șir de caractere terminat cu punct, reprezentând fraza
Liniile 2...: cuvant
• următoarele linii conțin câte un cuvânt din vocabular; fișierul se termină cu o linie liberă.

Date de ieșire
Fișier de ieșire: **TEXTMARE.OUT**
Linia 1: fraza
• șir de caractere terminat cu punct, reprezentând fraza despărțită în cuvinte; între oricare două cuvinte consecutive va apărea exact câte un blanc.

Restricții
• dacă există mai multe soluții, la ieșire va fi scrisă una singură;
• dacă nu există soluție, în fișierul de ieșire se va scrie doar cifra 0 (zero);
• într-o frază un cuvânt poate să apară de mai multe ori, fiecare apariție a sa fiind numărată.

Exemplu

TEXTMARE.IN	TEXTMARE.OUT
acestaesteuntext. text acesta acest a care este un	acesta este un text.

Timp de execuție: 2 secunde/test

Barajul de selecție

P050119: Deșert

Radu Ștefan, Brașov

Se dă o imagine alb-negru care reprezintă fotografia, realizată din satelit, a unei zone dintr-un deșert.

În această zonă se caută o construcție secretă de formă dreptunghiulară. Imaginea porțiunii de deșert analizate a

fost codificată într-o matrice având dimensiunile maxime de $N \times 255$ elemente. Imaginea construcției este codificată într-o matrice având $K \times 32$ elemente. Elementele celor două matrice pot fi numai caracterele `'.'` și `'#'`.

Cerință
Determinați de câte ori se regăsește fotografia construcției pe hartă.

Date de intrare
Fișier de intrare: **DESERT.IN**
Linia 1: `N K`
- două numere naturale nenule, N reprezentând numărul de linii al matricei care codifică fotografia deșertului, iar K numărul de linii al matricei pătratice care codifică fotografia construcției;

Liniile 2...*K*+1: $c_{i,1}\ c_{i,2}\ \ldots\ c_{i,32}$
- aceste K linii conțin în formă codificată matricea fotografiei construcției (32 de caractere `'#'` și `'.'` neseparate prin spații);

Liniile *K*+2...*K*+N+1: $d_{i,1}\ d_{i,2}\ \ldots\ d_{i,255}$
- aceste N linii conțin în formă codificată matricea fotografiei deșertului: (255 de caractere `'#'` și `'.'` neseparate prin spații).

Date de ieșire
Fișier de ieșire: **DESERT.OUT**
Linia 1: `NR`
- număr natural, reprezentând numărul de potriviri ale matricei fotografiei construcției în matricea fotografiei deșertului.

Restricții
- $3 \le N \le 1024$
- $1 < K < N$
- Fotografia construcției nu se va roti și nu se va oglindi.

Exemplu

DESERT.IN	DESERT.OUT
3 2	2
#....(*în total* 31 *de puncte*)....	
#....(*în total* 31 *de puncte*)....	
#....(*în total* 254 *de puncte*)....	
#....(*în total* 254 *de puncte*)....	
#....(*în total* 254 *de puncte*)....	

Timp de execuție: 1 secundă/test

P050120: Potrivire

prof. Tudor Sorin, București

Se consideră două numere naturale nenule N și S.

Cerință
Determinați numerele distincte $x_1, x_2, ..., x_N$ aparținînd mulțimii $\{1, 2, ..., N\}$ astfel încât

$$1 \cdot x_1 + 2 \cdot x_2 + ... + N \cdot x_N = S.$$

Date de intrare
Fișier de intrare: **POTRIV.IN**
Linia 1: `N S`
- două numere naturale nenule, separate printr-un spațiu, reprezentând numărul dat, respectiv suma dată.

Date de ieșire
Fișier de ieșire: **POTRIV.OUT**
Linia 1: $x_1\ x_2\ \ldots\ x_N$
- N numere naturale nenule, separate prin câte un spațiu, reprezentând soluția problemei. Dacă nu există soluție, pe această linie se va scrie numărul `0`.

Restricții
- $3 < N \le 1000$
- $0 < S \le 333834000$
- dacă există mai multe soluții, în fișier se va scrie una singură.

Exemplu

POTRIV.IN	POTRIV.OUT
4 26	3 2 1 4

Explicație
`1·3 + 2·2 + 3·1 + 4·4 = 26`

Timp de execuție: 1 secundă/test

P050121: Tort

Mihai Stroe, București

Gigel a primit de ziua lui un tort dreptunghiular pe care mama lui a desenat un caroiaj. În mijlocul anumitor pătrățele sunt așezate bezele, una într-un pătrățel. Gigel ar vrea să mănânce cât mai multe bezele, dar tatăl lui a inventat o șmecherie prin care speră să reducă pofta de bezele a lui Gigel.

Gigel a primit și o mulțime de bucăți de ciocolată. Tatăl lui îi cere să înlocuiască bezelele pe care vrea să le mănânce cu bucăți de ciocolată, formate din două pătrățele. Gigel trebuie să înlocuiască întotdeauna câte două bezele (vecine pe orizontală sau verticală), fără a suprapune bucățile de ciocolată.

Cerință
Știind că există suficiente bucăți de ciocolată, ajutați-l pe Gigel să înlocuiască cât mai multe bezele, respectând regulile impuse de tatăl lui.

Date de intrare
Fișier de intrare: **TORT.IN**
Linia 1: `M N`
- două numere naturale nenule, separate printr-un spațiu, reprezentând dimensiunile caroiajului de pe tort;

Liniile 2...*M*+1: $t_{i1}\ t_{i2}\ \ldots\ t_{iN}$
- aceste M linii conțin codificarea caroiajului de pe tort, linie după linie: valoarea 1 reprezintă o bezea, `0` înseam-

nă că în pătrățelul respectiv nu există bezea; între două valori numerice există un singur spațiu.

Date de ieșire
Fișier de ieșire: **TORT.OUT**
Linia 1: `K`
- număr natural, reprezentând numărul bucăților de ciocolată (o bucată este formată din două pătrățele, deci va înlocui două bezele); acest număr trebuie să fie cel mai mare posibil, respectând cerințele problemei; o bucată de ciocolată, formată din două pătrățele nu se poate rupe;

Liniile 2...*K*+1: X_i Y_i D_i
- aceste *K* linii conțin câte două numere naturale nenule și un caracter, separate prin câte un spațiu, care descriu amplasarea unei bucăți de ciocolată; X_i și Y_i sunt coordonatele pătrățelului stânga sus, iar D_i identifică poziția celuilalt pătrățel al bucății astfel: este `'E'` (*Est*) sau `'S'` (*Sud*), după cum al doilea pătrățel se află la dreapta primului sau sub primul.

Restricții
- $1 < M,N \leq 100$
- o bucată de ciocolată trebuie să înlocuiască exact două bezele
- bucățile de ciocolată nu se pot suprapune
- ordinea în fișierul de ieșire a descrierii bucăților de ciocolată poate fi oarecare.

Exemplu

TORT.IN	TORT.OUT
4 3	3
1 0 1	2 2 S
0 1 0	3 1 S
1 1 0	4 2 E
1 1 1	

Timp de execuție: 1 secundă/test

P050122: Descompuneri

șef lucrări Stelian Ciurea, Sibiu

Se dau numerele naturale nenule *N*, *K* și *X*. Numim o *K*-descompunere a lui *N* un șir strict crescător de *K* numere naturale nenule, a căror sumă este *N*.

Cerință
Dintre toate *K*-descompunerile lui *N*, să se determine în câte apare numărul *X*.

Date de intrare
Fișier de intrare: **DESCOMP.IN**
Linia 1: `N K X`
- trei numere naturale nenule, separate prin câte un spațiu, având semnificația din enunț.

Date de ieșire
Fișier de ieșire: **DESCOMP.OUT**
Linia 1: `NR`
- număr natural, reprezentând numărul descompunerilor respectând cerințele problemei.

Restricții
- $2 \leq N \leq 350$

Exemplu

DESCOMP.IN	DESCOMP.OUT
12 3 2	3

Explicații
12=1+**2**+9
12=**2**+3+7
12=**2**+4+6

Timp de execuție: 1 secundă/test

P050123: James Bond

Mihai Stroe, București

Un grup de teroriști s-a ascuns într-un sistem de canalizare. James Bond, având la dispoziție o hartă care descrie configurația sistemului de canalizare și pozițiile în care teroriștii au amplasat bombe, vrea să anihileze grupul de teroriști. Conductele din acest sistem sunt desenate pe hartă sub forma unor segmente de dreaptă în plan, astfel încât oricare două segmente au cel mult un punct comun, care este capăt pentru amândouă segmentele.

Explozia unei bombe afectează o zonă circulară (interior și frontieră) și este instantanee pe întreaga zonă afectată. Bombele pot să difere una de cealaltă prin raza de acțiune sau prin timpul la care explodează.

În urma unei explozii, canalele care trec prin zona de acțiune a bombei devin impracticabile pe segmentul afectat, dar, dacă Bond se află într-un astfel de canal și a trecut deja de zona afectată de explozie, poate să-și continue drumul.

Dacă James Bond se află în raza de acțiune a unei bombe la momentul exploziei, el moare.

La momentul inițial (momentul `0`), James Bond pornește dintr-un punct al sistemului de canalizare (capăt al unui segment) și se deplasează cu viteză constantă, egală cu 1. James Bond trebuie să ajungă în punctul în care se află teroriștii (capăt al unui segment) în cel mai scurt timp posibil.

Cerință
Determinați calea pe care trebuie să o urmeze James Bond astfel încât să ajungă viu la teroriști în cel mai scurt timp.

Date de intrare
Fișier de intrare: **BOND.IN**
Linia 1: `M N`
- două numere întregi, pozitive, separate printr-un spațiu, reprezentând numărul de conducte și numărul de bombe;

Liniile 2...*M*+1: X_1 Y_1 X_2 Y_2
- aceste *M* linii conțin fiecare câte patru numere întregi, separate prin câte un spațiu, reprezentând descrierea segmentelor prin coordonatele extremităților;

Liniile *M*+2...*M*+*N*+1: X_C Y_C R T
- aceste *N* linii conțin fiecare câte patru numere întregi, separate prin câte un spațiu, reprezentând descrierea bombelor prin coordonatele punctului unde sunt amplasate, raza cercului pe care acționează și momentul de timp la care acționează;

Linia *M*+*N*+2: X_P Y_P
- două numere întregi, separate printr-un spațiu, reprezentând coordonatele punctului de plecare, capăt al unui segment;

Linia *M*+*N*+3: X_D Y_D
- două numere întregi, separate printr-un spațiu, reprezentând coordonatele punctului în care se află teroriștii, capăt al unui segment.

Date de ieșire

Fișier de ieșire: **BOND.OUT**

Linia 1: TMIN
- număr real pozitiv, reprezentând timpul minim în care James Bond ajunge la teroriști;

Liniile 2...: C_1 C_2
- pe următoarele linii, până la sfârșitul fișierului, se vor scrie câte două numere întregi, separate printr-un spațiu, reprezentând coordonatele capetelor de segmente prin care va trece James Bond pentru a ajunge la grupul de teroriști.

Restricții

- $1 \le M \le 100$
- $0 \le N \le 50$
- $1 \le R \le 30000$
- $0 \le T \le 30000$
- coordonatele capetelor segmentelor și ale pozițiilor bombelor aparțin intervalului [-30000, 30000].

Exemplu

BOND.IN
```
3 1
0 0 10 0
0 0 10 10
10 0 10 10
6 4 2 1
0 0
10 10
```

BOND.OUT
```
20
0 0
10 0
10 10
```

Observații

- numerele din fișierul de ieșire se vor afișa cu o precizie de două zecimale;
- se garantează existența soluției;
- datele de intrare sunt corecte.

Timp de execuție: 1 secundă/test

P050124: Șiruri

Radu Ștefan, Brașov

Se dau două șiruri având ambele câte *N* numere naturale, cuprinse între 1 și *N*/2 (*N* este par). Fiecare număr între 1 și *N*/2 apare de exact două ori în fiecare dintre cele două șiruri.

Cerință

Determinați subșirul comun având lungimea maximă.

Date de intrare

Fișier de intrare: **SIRURI.IN**

Linia 1: N
- număr natural nenul având semnificația din enunț;

Linia 2: a_1 a_2 a_3 ... a_N
- *N* numere naturale nenule, separate prin câte un spațiu, reprezentând elementele primului șir;

Linia 3: b_1 b_2 b_3 ... b_N
- *N* numere naturale nenule, separate prin câte un spațiu, reprezentând elementele celui de-al doilea șir.

Date de ieșire

Fișier de ieșire: **SIRURI.OUT**

Linia 1: NR
- numărul de elemente ale subșirului comun;

Linia 2: c_1 c_2 c_3 ... c_{NR}
- *NR* numere naturale nenule, reprezentând subșirul comun de dimensiune maximă.

Restricții

- $0 < N \le 15000$
- *N* este număr par

Exemplu

SIRURI.IN
```
10
3 5 4 4 2 5 2 3 1 1
1 4 2 2 3 4 5 5 3 1
```

SIRURI.OUT
```
5
4 2 2 3 1
```

Timp de execuție: 1 secundă/test