Finala ACM

Pe data de 31 martie a avut loc la Praga etapa finală a concursului de programare pe echipe destinat studenților ACM. La această competiție au participat 73 de echipe de studenți din toată lumea. Aceste echipe au fost selectate dintr-un număr de 3150 de echipe din 1411 universități în urma fazelor regionale ale concursului. În continuare vă prezentăm versiunea în limba română, realizată de redacția GInfo, a celor 10 probleme propuse spre rezolvare la acest concurs.

P040401: Carl Furnica

Furnicile lasă urme chimice pe pământ pentru a marca drumul pe care îl străbat pentru ca alte furnici să le urmeze. De obicei, aceste urme marchează un drum drept, dar într-o colonie de furnici există o furnică puțin neobișnuită pe nume *Carl*. *Carl* va merge destul de des în zig-zag fără nici un motiv, uneori intersecându-și traseul de mai multe ori. Când alte furnici ajung la o intersecție, ele urmează întotdeauna drumul cu cel mai puternic miros, care este și cel mai recent drum care indică furnicilor în ce direcție să părăsească intersecția respectivă.

Furnicile au lungimea de un centimetru, înaintează cu viteza de 1 cm/s și urmează cu exactitate drumurile (când ocolesc un colț, descriu unghiuri de 90 de grade). Furnicile nu se pot călca sau depăși unele pe altele.

Dacă două furnici ajung într-un punct dintr-o intersecție în același timp, atunci are prioritate de trecere cea care a parcurs cel mai lung drum făcut de *Carl*. Dacă nu există o astfel de furnică, atunci are prioritate cea care a așteptat cel mai mult timp în intersecție.

Carl iese de sub pământ în originea sistemului de coordonate la momentul de timp $t = 0$ secunde. Apoi el construiește drumul, iar la sfârșit intră sub pământ. Restul furnicilor urmează drumul la intervale regulate de timp.

Dându-se descrierea drumului lui *Carl* și momentele când restul furnicilor urmează drumul, trebuie să determinați în cât timp mulțimea de furnici va termina de parcurs drumul. Se garantează faptul că toate furnicile vor parcurge drumul în întregime.

Date de intrare

Fișierul de intrare **`carl.in`** conține mai multe cazuri de test. Prima linie a fișierului de intrare conține un singur număr întreg care reprezintă numărul de teste care urmează.

Datele fiecărui test sunt scrise pe mai multe linii. Astfel, prima linie a unui test conține trei numere întregi pozitive n, m și d, separate între ele printr-un spațiu, care reprezintă numărul de segmente ale drumului, numărul de furnici care urmează drumul (inclusiv *Carl*), respectiv intervalul de timp la care o furnică iese din pământ și urmează drumul.

Carl pornește la momentul de timp 0, următoarea furnică pornește după d secunde, următoarea după $2 \cdot d$ secunde ș.a.m.d. Atunci când canalul de ieșire este blocat, furnicile vor ieși din pământ cât mai repede posibil, în ordinea corectă.

Fiecare dintre următoarele n linii ale unui test conține două numere întregi x și y, separate între ele printr-un singur spațiu, care reprezintă coordonatele capătului unui segment din drumul lui *Carl*, în ordinea în care *Carl* construiește drumul.

Originea primului segment este dată de punctul de coordonate (0, 0), iar originea oricărui alt segment este dată de capătul segmentului precedent.

Pentru simplitate, *Carl* călătorește tot timpul pe segmente paralele cu axele de coordonate și nici un capăt al unui segment nu se află pe alt segment decât dacă este unul dintre capetele acestuia din urmă.

Date de ieșire

Pentru fiecare test rezultatele trebuie furnizate la *ieșirea standard*, după descrierea următoare:

```
Case C:
Carl finished the path at time t1
The ants finished in the following order:
```

```
a1 a2 a3 ... am
The last ant finished the path at time t2
```
unde `C` reprezintă numărul testului (începând cu 1); `a1`, `a2`, ..., `am` reprezintă numerele de ordine ale furnicilor în ordinea în care acestea termină de parcurs drumul și intră înapoi în pământ; `t1` și `t2` reprezintă momentele de timp la care *Carl* și ultima furnică termină de parcurs drumul.

Rezultatele furnizate pentru teste trebuie separate între ele printr-o linie vidă.

Restricții și precizări

- $1 \le n \le 50$;
- $1 \le m, d \le 100$;
- coordonatele care reprezintă capetele segmentelor drumului lui *Carl* sunt numere întregi cuprinse între -100 și 100.

Exemplu

carl.in

```
2
4 7 4
0 4
2 4
2 2
-2 2
4 7 2
0 4
2 4
2 2
-2 2
```

Ieșire standard

```
Case 1:
Carl finished the path at time 13
The ants finished in the following order:
0 2 1 3 4 5 6
The last ant finished the path at time 29

Case 2:
Carl finished the path at time 13
The ants finished in the following order:
0 4 1 5 2 6 3
The last ant finished the path at time 19
```

P040402: Heliport

În această eră a vitezei, companiile investesc în heliporturi pentru a reduce timpul călătoriilor membrilor conducerii. Heliporturile sunt de obicei zone de aterizare circulare plasate pe acoperișurile sediilor principale ale companiilor.

Sarcina voastră este să scrieți un program care determină raza cea mai mare a unui heliport circular care poate fi construit pe acoperișul unei clădiri care are forma unui poligon simplu.

Deoarece aceasta este doar faza de proiectare a construcției, programul trebuie să determine doar raza heliportului.

Date de intrare

Fișierul de intrare **heliport.in** conține mai multe teste.

Datele pentru fiecare test constau în două linii. Prima linie conține un număr întreg *n* care reprezintă numărul de laturi ale poligonului care descrie forma acoperișului clădirii. Cea de-a doua linie conține *n* perechi de forma (*m*, *d*).

Presupunând că acoperișul este desenat în plan, *m* este un număr întreg care reprezintă lungimea unei laturi și *d* este un caracter care reprezintă direcția în care se va desena această latură. Desenarea laturilor se va face în sens trigonometric pornind de la punctul în care s-a ajuns cu desenatul la pasul anterior. Caracterul *d* poate avea valorile *U*, *R*, *D* sau *L* care semnifică faptul că latura se va desena în sus, la dreapta, în jos sau la stânga.

Segmentele care reprezintă laturile poligonului sunt paralele cu axele de coordonate și sunt date în ordine trigonometrică. Poziția de început a desenului este dată de punctul de coordonate (0, 0).

După ultimul test din fișierul de intrare se va afla valoarea 0 pe o singură linie.

Date de ieșire

Pentru fiecare test din fișierul de intrare se va tipări la *ieșirea standard*, pe o linie numărul de ordine al acestuia (începând cu 1) și un număr real rotunjit la două zecimale exacte care reprezintă raza heliportului care poate fi construit.

Liniile furnizate la *ieșirea standard* trebuie separate între ele printr-o linie vidă și trebuie să aibă formatul:

```
Case Number # radius is: r
```

unde `#` reprezintă numărul testului, iar `r` este raza determinată pentru heliportul de la testul respectiv.

Restricții și precizări

- $4 \le n \le 20$;
- lungimea unei laturi a poligonului care descrie forma acoperișului clădirii este un număr întreg cuprins între 1 și 50.

Exemplu

heliport.in

```
4
2 R 2 U 2 L 2 D
10
10 R 10 U 10 L 10 U 10 R 5 U 30 L 20 D 20 R 5 D
0
```

Ieșire standard

```
Case Number 1 radius is: 1.00

Case Number 2 radius is: 10.00
```

Explicație

În figura 1 se poate observa harta acoperișului clădirii pentru cel de-al doilea set de date, iar raza maximă a heliportului este 10.

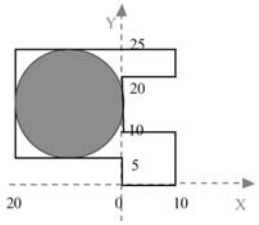

Figura 1

P040403: Imaginea este totul

Compania la care lucrați construiește roboți care pot transporta obiecte ușoare. Un astfel de robot trebuie să aibă inteligența necesară pentru a determina dacă un anumit obiect este suficient de ușor pentru a putea fi transportat. Acest lucru se realizează prin fotografierea obiectului din cele șase direcții nord, sud, est, vest, sus și jos și apoi, prin calcularea unei limite superioare a greutății obiectului în funcție de aceste imagini.

Sarcina voastră este să scrieți un program care realizează acest lucru pentru robot. Se poate presupune că fiecare obiect este format din cubulețe de dimensiune $1 \times 1 \times 1$ dispuse într-o matrice tridimensională de dimensiune $N \times N \times N$. Din această matrice tridimensională unele cubulețe pot lipsi. Fiecare cubuleț de dimensiune $1 \times 1 \times 1$ are greutatea de un gram și este desenat cu o singură culoare.

Obiectul nu este neapărat conex.

Date de intrare

Datele de intrare se citesc din fișierul **`image.in`** care conține mai multe teste. Pentru fiecare test se vor citi mai multe linii. Pe prima linie corespunzătoare unui test se află un număr natural N care reprezintă dimensiunile cubului.

Următoarele N linii conțin cele șase imagini de dimensiune $N \times N$ în ordinea *față*, *stânga*, *spate*, *dreapta*, *sus* și *jos*. Pe fiecare dintre aceste N linii se află șase grupuri, separate între ele prin spațiu, a câte N caractere fiecare. Un grup reprezintă o linie a uneia dintre cele șase imagini. Caracterele imaginilor pot fi litere mari ale alfabetului englezesc sau caracterul "`.`"(punct). Literele mari descriu culoarea cubulețului respectiv, iar caracterul "`.`" indică faptul că în zona respectivă este spațiu liber.

După ultimul test se va afla valoarea `0` pe o singură linie.

Date de ieșire

La *ieșirea standard* se vor afișa greutățile obiectelor în ordinea în care acestea au fost citite folosind formatul: `Maximum weight: # gram(s)`, unde `#` reprezintă greutatea maximă pe care o poate avea un obiect.

Restricție

- $1 \le N \le 10$;

Exemplu

`image.in`

```
3
.R. YYR .Y. RYY .Y. .R.
GRB YGR BYG RBY GYB GRB
.R. YRR .Y. RRY .R. .Y.
2
ZZ ZZ ZZ ZZ ZZ ZZ
ZZ ZZ ZZ ZZ ZZ ZZ
0
```

Ieșire standard

```
Maximum weight: 11 gram(s)
Maximum weight: 8 gram(s)
```

P040404: Nesiguranță în Praga

Praga este un oraș periculos pentru dezvoltatorii de scheme criptografice. În anul 2001 o echipă de cercetători din *Praga* au anunțat câteva lipsuri ale faimosului protocol de criptare ***PGP*** (***P**retty **G**ood **P**rivacy*). În anul 2003 au fost descoperite lipsuri ale protocoalelor ***SSL*** și ***TLS*** (***S**ecure **S**ockets **L**ayer* și ***T**ransport **L**ayer **S**ecurity*). Cu toate acestea, reputația *Pragăi* referitoare la protocoalele criptografice nu i-a împiedicat pe criptograful amator și încăpățânatul *Immanuel Kant-DeWitt* (cunoscut de prietenii săi ca *I. Kant-DeWitt*) să-și aducă ultima sa schemă de criptare la *Praga*.

Această schemă funcționează astfel: când se transmite un mesaj p de lungime n, transmițătorul alege un număr întreg $m \ge 2 \cdot n$ și numerele întregi s, t, i și j, unde $0 \le s, t, i, j < m$ și $i < j$. m reprezintă lungimea textului cifrat c care va fi transmis.

Inițial c conține m locuri libere. Prima literă a mesajului p va fi scrisă pe poziția s din c. A k-a ($k \ge 2$) literă din p va fi pusă în c cu i poziții libere mai departe de cea în care a fost pusă cea de-a (k-1)-a literă a lui p, numărând poziții și de la începutul lui c dacă este necesar. Pozițiile care conțin litere nu sunt considerate a fi libere.

De exemplu, dacă mesajul este `PRAGUE` și $s = 1$, $i = 6$ și $m = 15$ atunci literele sunt plasate în c astfel:

A	P	_	U	_	_	_	_	R	G	_	_	E	_	_
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Începând cu poziția t sau prima poziție liberă începând căutarea cu prima poziție din dreapta lui t, mesajul p mai este introdus încă o dată în c, dar de data aceasta numărând câte j poziții între litere.

De exemplu, dacă $t = 0$ și $j = 8$ a doua copie a lui p este inserată astfel (începând cu poziția 2, care este prima poziție liberă de la $t = 0$):

A	P	P	U	R	_	A	U	R	G	E	G	E	_	_
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

În final, pozițiile care rămân libere sunt ocupate cu litere alese aleator:

A	P	P	U	R	A	A	U	R	G	E	G	E	W	E
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Kant-DeWitt este de părere că duplicarea mesajului combinată cu utilizarea de litere aleatoare va păcăli sche-

mele de decriptare care se bazează pe frecvența de apariție a literelor și că, dacă nu se cunosc valorile *s* și *i*, nimeni nu poate afla care este mesajul original.

Sarcina voastră este să îi demonstrați lui *Kant-DeWitt* că nu are dreptate. Dându-se un număr de texte cifrate (și nici o altă informație adițională), determinați cel mai lung mesaj care se poate codifica folosind metoda lui *Kant-DeWitt*.

Date de intrare

Fișierul de intrare **`insecure.in`** conține mai multe mesaje cifrate, câte unul pe linie. Fiecare mesaj este format din litere mari ale alfabetului englezesc și nu este precedat sau succedat de spații. Fiecare mesaj va avea lungimea cuprinsă între 1 și 40 de caractere.

Ultimul mesaj cifrat va fi urmat de o linie pe care se va afla litera `X`.

Date de ieșire

La *ieșirea standard*, pentru fiecare mesaj cifrat se va afișa numărul de ordine al acestuia și cel mai lung șir de caractere care ar fi putut fi criptat sub forma prezentă în fișierul de intrare. Dacă există mai mult de un șir de caractere de lungime maximă care pot fi codificate la fel, va trebui să tipăriți fraza `"Codeword not unique"`.

Fiecare linie din fișierul de ieșire va avea formatul:

```
Code #: s
```

sau

```
Code #: Codeword not unique
```

unde `#` reprezintă numărul de ordine al mesajului cifrat în fișierul de intrare, iar `s` reprezintă mesajul original.

Exemplu

`insecure.in`

```
APPURAAURGEGEWE
ABABABAB
THEACMPROGRAMMINGCONTEST
X
```

Ieșire standard

```
Code 1: PRAGUE
Code 2: Codeword not unique
Code 3: Codeword not unique
```

P040405: Date care se suprapun

Un grup de cercetători dezvoltă un program pe calculator care va extrage date mai vechi, legate de cotațiile bursiere de pe piață, folosind un serviciu care cere o taxă fixă pentru fiecare cotație bursieră pe care o furnizează.

Grupul a analizat colecția de cotații extrase anterior și a descoperit multe duplicate, de unde rezultă că au fost irosiți bani.

Așadar, noul program va trebui să mențină o listă a cotațiilor cerute anterior de membrii grupului iar în momentul în care sunt necesare alte cotații, va extrage doar cotațiile din datele pentru care nu există informații, minimizând astfel costul.

Sarcina voastră este să scrieți un program care determină când trebuie extrase cotații noi. Datele de intrare pentru program vor consta în intervale de timp pentru care, în trecut, au fost cerute cotații bursiere și intervale de timp pentru care sunt necesare noi cotații. Programul va determina intervalele de timp pentru care trebuie extrase cotații folosind serviciul cu taxă.

Date de intrare

Fișierul de intrare **`intersect.in`** va conține mai multe seturi de date.

Fiecare set de date este descris pe mai multe linii. Prima linie conține două numere întregi *NX* și *NR*, separate între ele printr-un singur spațiu, care reprezintă numărul de intervale de timp pentru care există informații, respectiv numărul de intervale de timp pentru care sunt necesare informații. Următoarele *NX* + *NR* linii conțin perechi de date. Prima dată dintr-o pereche este mai mică sau egală cu cea de-a doua. Primele *NX* perechi de date reprezintă intervalele de timp pentru care există informații, iar restul de *NR* perechi reprezintă intervalele de timp pentru care sunt necesare informații legate de cotațiile bursiere. Cele două date dintr-o pereche sunt separate între ele printr-un singur spațiu.

Ultimul set de date din fișierul de intrare este urmat de o linie pe care se află două valori `0` separate între ele printr-un singur spațiu.

Fiecare dată din fișierul de intrare va avea formatul *YYYYMMDD*, unde *YYYY* reprezintă anul (care este cuprins între 1700 și 2100), *MM* reprezintă luna (care poate avea valorile 01, 02, ..., 12), iar *DD* reprezintă ziua (care reprezintă o zi validă în raport cu anul și luna). Ziua poate avea valoarea minimă 01, valoarea maximă 31 pentru lunile 01, 03, 05, 07, 08, 10 și 12, valoarea maximă 30 pentru lunile 04, 06, 09 și 11, valoarea maximă 28 pentru luna 02, dacă anul nu este bisect, și valoarea maximă 29 pentru luna 02, dacă anul este bisect. Un an este bisect dacă este divizibil cu 4 și nu este divizibil cu 100 sau dacă este divizibil cu 400.

Date de ieșire

La ieșirea standard, pentru fiecare set de date din fișierul de intrare se va scrie numărul de ordine al acestuia pe o linie urmată de mai multe linii pe care se vor scrie intervalele de timp pentru care trebuie extrase informații sau o linie pe care se va afla textul `No additional quotes are required`, ca în exemplu. Rezultatele furnizate pentru fiecare set de date vor fi separate între ele printr-o linie vidă.

Restricții

- $0 \le NX, NR \le 100$.

Exemplu

`intersect.in`

```
1 1
19900101 19901231
```

```
19901201 20000131
0 3
19720101 19720131
19720201 19720228
19720301 19720301
1 1
20010101 20011231
20010515 20010901
0 0
```

Ieșire standard

```
Case 1:
    1/1/1991 to 1/31/2000

Case 2:
    1/1/1972 to 2/28/1972
    3/1/1972

Case 3:
    No additional quotes are required.
```

P040406: Alipind hărți

Fotografiile unei zone, luate din avion sau satelit, pentru care trebuie realizată o hartă, au de obicei o rezoluție mare pentru a identifica în mod unic caracteristicile majore pe care aceasta la deține. Din moment ce o fotografie acoperă doar o mică porțiune a pământului, realizarea unei hărți pentru o zonă mai mare necesită mai multe fotografii care se suprapun pe porțiuni și prin alipirea fotografilor rezultă o hartă a unei zone mai mari.

Pentru această problemă aveți la dispoziție câteva hărți dreptunghiulare, fiecare fiind reprezentat sub forma unui tablou bidimensional de caractere. O celulă poate conține o literă majusculă a alfabetului englezesc în cazul în care corespunde unei zone care conține o caracteristică care identifică în mod unic zona. Litere diferite corespund unor caracteristici diferite, dar aceeași caracteristică poate identifica mai multe celule (de exemplu, o șosea). O celulă conține caracterul '-' dacă în locul respectiv nu există o caracteristică importantă. Alipirea a două hărți constă în suprapunerea lor astfel încât una sau mai multe caracteristici importante să fie suprapuse. O celulă care conține o caracteristică importantă pe una din hărți, poate fi suprapusă peste o celulă, de pe altă hartă, care nu conține o caracteristică importantă, dar nu pot fi suprapuse două celule care conțin caracteristici importante distincte.

```
          --A-C  C----  C----  ----D  -D--C
          ----D  D---F  -----  -E--B  ----G
          ----B  B----  B-A-C  -----  ----B
Harta nr.:  1      2      3      4      5
```

Dacă se consideră hărțile de mai sus care au trei linii și cinci coloane fiecare, se poate observa că cea mai din dreapta coloană a primei hărți se potrivește perfect cu prima coloană a celei de-a doua hărți. Așadar, prima și a doua hartă pot fi suprapuse și rezultă o hartă care are trei linii și nouă coloane. Prima hartă se mai poate suprapune și cu harta a treia din moment ce caracteristicile importante `B` și `C` de pe ultima coloană a primei hărți se potrivesc cu cele de pe prima coloană a celei de-a treia hărți; caracteristica `D` nu se potrivește perfect peste caracterul '-', dar nu există nici un conflict. Similar, prima linie a hărții 1 se poate suprapune cu ultima linie a hărții 3.

"Scorul" unei perechi de hărți indică suprafața pe care se potrivesc cele două hărți. Scorul suprafeței suprapuse a două hărți reprezintă numărul de celule care conțin caracteristici importante, care coincid în suprapunere și care dau cea mai bună potrivire. Scorul unei perechi de hărți este dat de scorul maxim care se poate obține pentru o suprapunere validă. Așadar, scorul unei perechi de hărți care au fiecare trei linii și cinci coloane este un număr întreg cuprins între `0` și 15.

Un "deplasament" reprezintă o pereche de numere întregi (*r*, *c*) care indică modul în care două hărți *a* și *b* se pot suprapune. Valoarea *r* reprezintă deplasamentul liniilor din *b* relativ la liniile din *a*, iar valoarea *c* reprezintă deplasamentul coloanelor din *b* relativ la coloanele din *a*.

De exemplu, suprapunerea hărților 1 și 2 are deplasamentul (`0`, 4) și scorul 3, iar cele două suprapuneri posibile ale hărților 1 și 3 cu scorul 2 au deplasamentele (`0`, 4), respectiv (-2, `0`).

Următorii pași descriu modul în care se realizează alipirea unui secvențe de hărți:

- se suprapun hărțile care au cel mai mare scor pozitiv (se rezolvă conflictele folosind harta cu cel mai mic număr de ordine);
- se elimină hărțile care au fost suprapuse din secvență;
- se adaugă rezultatul suprapunerii celor două hărți în secvență, atribuindu-i-se cel mai mare număr de ordine care apare în secvență și la care se adaugă 1.

În exemplul anterior, hărțile 1 și 2 prin suprapunere formează harta cu numărul de ordine 6, iar hărțile 1 și 2 sunt eliminate din secvență. Cei trei pași descriși anterior se repetă până când secvența conține o singură hartă sau până când nu mai pot fi suprapuse hărți (scorul oricărei perechi de hărți din secvență este `0`).

Dacă două hărți se pot suprapune în mai multe feluri cu același scor, atunci ele trebuie suprapuse în funcție de cel mai mic deplasament pentru linii și apoi în funcție de cel mai mic deplasament pentru coloane.

Date de intrare

Fișierul de intrare **`maps.in`** conține mai multe seturi de date fiecare având între două și zece hărți.

Fiecare set de date constă în mai multe linii. Prima linie a unui set de date conține un singur număr întreg care reprezintă numărul de hărți pentru setul de date respectiv. În continuare urmează hărțile.

Fiecare hartă constă în mai multe linii. Prima linie a unei hărți conține două numere întregi *NR* și *NC*, separate între

ele printr-un singur spațiu, care reprezintă numărul de linii, respectiv numărul de coloane ale hărții respective. Următoarele *NR* linii ale descrierii unei hărți conțin câte o linie a hărții care este dată doar de primele *NC* caractere de pe ea, restul caracterelor suplimentare trebuind ignorate.

Ultimul set de date este urmat de o linie pe care se va afla valoarea `0`.

Date de ieșire

La ieșirea standard, pentru fiecare set de date trebuie afișat numărul său de ordine (începând cu 1) și hărțile obținute, fiecare identificată prin numărul său de ordine și trebuie mărginite de o frontieră. Furnizarea rezultatelor trebuie să se facă folosind formatul din exemplu. Nici o hartă nu trebuie să aibă mai mult de 70 de coloane.

Restricții

• $1 \le NR, NC \le 10$.

Exemplu

`maps.in`

```
5
3 5
--A-C
----D
----B
3 5
C----
D---F
B----
3 5
C----
-----
B-A-C
3 5
----D
-E--B
-----
3 5
-D--C
----G
----B
2
3 5
----A
----B
----C
3 5
A----
B----
D----
0
```

Ieșire standard

```
Case 1
     MAP 9:
     +-------------+
     |-D--C--------|
     |----G--------|
     |----B-A-C----|
     |----B-A-C----|
     |-----E--B----|
     |-------------|
     +-------------+

Case 2
     MAP 1:
     +-----+
     |----A|
     |----B|
     |----C|
     +-----+

     MAP 2:
     +-----+
     |A----|
     |B----|
     |D----|
     +-----+
```

P040407: Navigare

GPS-ul este un sistem de navigare care se bazează pe un set de sateliți care se află pe orbite la aproximativ 20.000 de km deasupra pământului. Fiecare satelit descrie o orbită cunoscută și transmite semnale radio care codifică timpul curent. Dacă un vehicul echipat cu un dispozitiv *GPS* are un ceas precis, poate compara timpul său local cu cel codificat de semnalele recepționate de la sateliți. Deoarece semnalele radio sunt transmise cu o frecvență cunoscută, vehiculul poate determina distanța dintre poziția sa și poziția satelitului în momentul în care a transmis semnalul radio. Măsurând distanța sa față de mai mulți sateliți de pe orbite cunoscute, un vehicul poate să-și determine poziția cu o precizie foarte mare.

Figura 2

Sarcina voastră este să scrieți un program simplu de "pilot-automat" care se bazează pe navigare *GPS*. Pentru ca problema să fie mai simplă, vom presupune că totul se desfășoară în plan. Cu alte cuvinte, nu trebuie luată în considerare curbura pământului sau altitudinea sateliților. Mai mult, în problemă se folosesc viteze care sunt mai apropiate celor ale avioanelor și undelor sonore decât cele ale sateliților și undelor radio.

Dându-se un set de semnale ale unor surse mobile, programul vostru trebuie să calculeze poziția în care se recepționează semnalul în plan. Apoi, dându-se un punct de destinație în plan, programul trebuie să determine direcția indicată de o busolă și care trebuie urmată pentru a ajunge din punctul determinat până în punctul destinație. Toate direcțiile busolei sunt în grade. Gradul `0` corespunde direcției nordului, gradul 90 corespunde direcției est, gradul 180 corespunde direcției sud și gradul 270 corespunde direcției vest, după cum se poate observa și în figura 2.

Date de intrare

Fișierul de intrare **`navigation.in`** conține mai multe seturi de date.

Prima linie a fiecărui set de date conține un număr întreg pozitiv *N*, care reprezintă numărul de surse care emit semnale, urmat de trei numere reale *t*, *x* și *y* care reprezintă momentul de timp local exprimat în secunde când semnalele au fost primite relativ la momentul de timp `0`, iar *x* și *y* reprezintă coordonatele punctului destinație. Numerele de pe această linie sunt separate între ele printr-un spațiu.

Fiecare dintre următoarele *N* linii conține patru numere reale, separate între ele printr-un spațiu, care conțin informațiile legate de o sursă. Primele două numere reprezintă poziția sursei la momentul de timp `0`. Al treilea număr re-

prezintă direcția de deplasare *D* a sursei exprimată în grade, iar al patrulea număr reprezintă timpul codificat de sursă.

Ultimul set de date este urmat de o linie pe care se află patru valori 0 separate între ele printr-un spațiu.

Date de ieșire

La ieșirea standard, pentru fiecare set de date se va afișa pe o linie numărul de ordine al setului din fișierul de intrare și direcția în grade care trebuie urmată pentru a se ajunge la destinație, cuvântul `"Inconclusive"`, dacă direcția nu poate fi determinată, cuvântul `"Inconsistent"`, dacă datele sunt inconsistente sau cuvântul `"Arrived"`, dacă distanța dintre punctul în care se află vehiculul și punctul de sosire este mai mică sau egală cu 0,1 metri.

Fiecare linie furnizată la ieșirea standard poate avea forma:

```
Trial #: D degrees
Trial #: Inconclusive
Trial #: Inconsistent
```

sau

```
Trial #: Arrived
```

unde `#` reprezintă numărul setului de date, iar `D` reprezintă direcția de deplasare exprimată în grade.

Restricții și precizări

- $1 \le N \le 10$;
- numerele din fișierul de intrare nu sunt mai mari decât 10.000 și nici un număr real nu are mai mult de 5 zecimale;
- toate numerele care reprezintă coordonate sunt date în metri;
- sursele care emit semnal se deplasează cu 100 de metri pe secundă, iar semnalele se propagă cu 350 de metri pe secundă;
- datorită pierderii preciziei în momentul sincronizării ceasurilor, distanțele se pot calcula cu o eroare de 0,1 metri, mai exact, dacă distanța dintre două puncte este mai mică sau egală cu 0,1 metri se poate considera că acestea sunt identice;
- există posibilitatea ca un semnal să fie corupt la transmitere, ceea ce conduce la faptul că datele mai multor semnale să fie inconsistente.

Exemplu

`navigation.in`

```
3 2.53571 1050.0 1050.0
-100.0 350.0 90.0 1.75
350.0 -100.0 0.0 1.75
350.0 800.0 180.0 1.75
2 2.0 1050.0 1050.0
-100.0 350.0 90.0 1.0
350.0 -100.0 0.0 1.0
0 0 0 0
```

Ieșire standard

```
Trial 1: 45 degrees
Trial 2: Inconclusive
```

Explicație

Figura 3 corespunde primului set de date din fișierul de intrare. Pozițiile celor 3 sateliți la momentul de timp $t = 0$ sunt A(-100, 350), B(350, -150) și C(350, 800). Semnalele recepționate de dispozitivul *GPS* au fost transmise la momentul $t = 1{,}75$ când sateliții erau în pozițiile A', B', respectiv C' (în general, semnalele nu sunt transmise în același timp de către toți sateliții). Semnalele de la cei trei sateliți converg către punctul D la momentul de timp $t = 2{,}53571$ ceea ce înseamnă că D reprezintă poziția vehiculului. Din punctul D trebuie urmată direcția de 45 de grade indicată de busolă pentru a ajunge în punctul destinație de coordonate (1050, 1050).

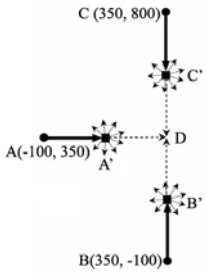

Figura 3

P040408: Copacii de pe străzi

Consiliul local al orașului *Greenville* a votat recent îmbunătățirea imaginii străzilor din oraș. Pentru a oferi mai multă verdeață peisajului, consiliul a decis să planteze copaci de-a lungul celor mai importante străzi și bulevarde. Pentru a putea aproxima costul acestei îmbunătățiri urbane, consiliul dorește să știe câți copaci vor fi plantați. Plantarea copacilor trebuie să respecte următoarele restricții:

- de-a lungul unei străzi copacii trebuie plantați la o distanță de minim 50 de metri unii de alții pentru a li se oferi o creștere și dezvoltare normală și pentru a menține costul acestei îmbunătățiri între niște limite rezonabile;
- din motive de siguranță nu trebuie plantați copaci la mai puțin 25 de metri de cea mai apropiată intersecție pentru ca participanții la trafic să se poată vedea unii pe alții cu ușurință când se apropie de intersecție. Siguranța traficului nu trebuie să fie compromisă de reducerea vizibilității.

Toate străzile considerate sunt drepte. Nu există curbe de-a lungul lor.

Consiliul orașului dorește să cunoască numărul maxim de copaci care pot fi plantați în conformitate cu cele două restricții.

Date de intrare

Fișierul de intrare **`streets.in`** conține mai multe hărți de străzi.

Prima linie a unei hărți conține un singur număr întreg n care reprezintă numărul de străzi ale hărții. Fiecare dintre următoarele n linii descrie o stradă ca un segment de dreaptă din plan. O linie a fișierului de intrare care descrie

o stradă conține patru numere întregi x_1, y_1, x_2 și y_2, separate între ele printr-un spațiu, cu semnificația că strada curentă merge din punctul de coordonate (x_1, y_1) până în punctul de coordonate (x_2, y_2).

Ultima descriere a unei hărți de străzi este urmată de o linie care conține valoarea `0`.

Date de ieșire

La ieșirea standard trebuie furnizate, pentru fiecare hartă, numărul de ordine al hărții din fișierul de intrare și numărul maxim de copaci care pot fi plantați astfel încât să fie respectate restricțiile din enunț.

Pentru fiecare hartă se vor scrie două linii. Prima linie trebuie să aibă formatul:

```
Map #
```

unde `#` reprezintă numărul de ordine al hărții din fișierul de intrare, începând cu 1 ca în exemplu, iar cea de-a doua linie trebuie să aibă formatul:

```
Trees = c
```

unde `c` reprezintă numărul maxim de copaci care pot fi plantați pe străzile care aparțin unei hărți.

Restricții și precizări

- $1 \le n \le 100$;
- coordonatele punctelor care descriu capetele unei străzi sunt numere întregi cuprinse între 0 și 1.000.000;
- toate străzile au lungimi pozitive și fiecare capăt aparține unei singure străzi;
- pentru fiecare stradă distanțele dintre două intersecții vecine nu sunt multipli exacți de 25 de metri; mai exact, diferența dintre o astfel de distanță și cel mai apropiat multiplu de 25 va fi cel puțin 0.001 metri.
- în fiecare intersecție se întâlnesc exact două străzi.

Exemplu

streets.in

```
3
0 40 200 40
40 0 40 200
0 200 200 0
4
0 30 230 30
0 200 230 200
30 0 30 230
200 0 200 230
3
0 1 121 1
0 0 121 4
0 4 121 0
0
```

Ieșire standard

```
Map 1
Trees = 13
Map 2
Trees = 20
Map 3
Trees = 7
```

P040409: Suspans!

Jan și *Tereza* locuiesc în clădiri vecine și apartamentele lor sunt față în față. Pentru proiectul de la disciplina științe de la școală ei doresc să construiască un pod suspendat din cablu, sfoară și carton care să facă legătura între cele două clădiri. Două bucăți identice ca lungime formează cablurile principale de suspensie și sunt atașate de marginea de jos a ferestrelor lor. Platforma de carton care reprezintă suprafața netedă a podului este susținută de mai multe sfori care sunt legate de cablurile principale de suspensie. Podul orizontal se află la exact un metru sub cel mai de jos punct al cablurilor de suspensie. Din motive estetice platforma podului ar trebui să fie la cel puțin doi metri sub cea mai de jos muchie a celei mai de jos ferestre a fiecăruia dintre cei doi elevi. Legile fizicii spun că fiecare cablu de suspensie are forma unei parabole.

Cu toate că *Jan* și *Tereza* nu au de gând să se plimbe pe acest prototip de pod, apare o problemă serioasă: unii locatari ai celor două imobile au pisici, iar alții au canari. Cei doi doresc să fie siguri că podul lor nu va putea permite unei pisici să ajungă la un canar. Cei doi au observat că o pisică nu poate sări pe ceva ce este mai înalt de 0,5 metri și nu poate sări de la o înălțime mai mare de trei metri. Deci, atâta timp cât podul se află la mai mult de 0,5 metri deasupra marginii de jos a geamului la care se află o pisică sau la cel puțin trei metri sub marginiea de jos a geamului la care se află o pisică, atunci aceasta nu va sări. De asemenea, dacă o pisică reușește să sară pe pod, atunci nu va putea ajunge la geamul unui canar dacă podul se află la mai mult de 0,5 metri sub marginea de jos a geamului la care se află canarul sau la cel puțin trei metri deasupra marginii de jos a geamului la care se află canarul. Pisicile sunt interesate doar să ajungă la canari și nu să se întoarcă acasă.

Figura de mai jos arată că apartamentul lui *Jan*, etichetat cu *J* și cel al *Terezei*, etichetat cu *T*, sunt legate printr-un cablu care unește punctele cele mai de jos ale ferestrelor lor și podul propriu-zis se află la un metru dedesubtul celui mai de jos punct al cablului de suspensie. Pisica de la etajul al doilea poate ajunge la canarul de la etajul al doilea din clădirea vecină folosind podul.

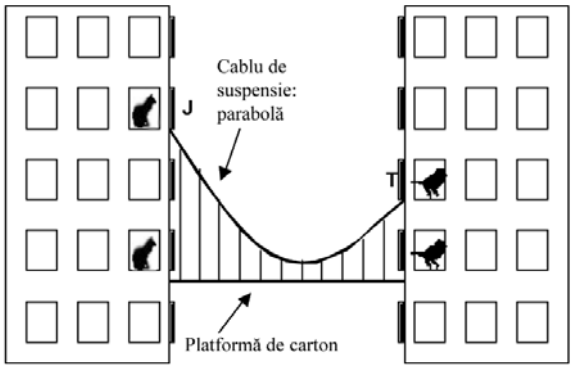

Figura 4

Sarcina voastră este să scrieți un program care determină de câtă sfoară au nevoie cei doi pentru a realiza fiecare segment de cablu al podului astfel încât să nu pună în pericol nici un canar din cele două clădiri.

Programul trebuie să țină cont de:
- distanța, exprimată în metri, dintre cele două imobile;
- numerele etajelor la care locuiesc cei doi elevi;
- tipul de animale care se află la etajele dedesubtul etajului la care stă *Jan*;
- tipul de animale care se află la etajele dedesubtul etajului la care stă *Tereza*.

Programul trebuie să determine lungimea celui mai lung cablu care poate fi folosit la suspendarea podului care leagă cele două clădiri și care nu permite nici unei pisici să ajungă la un canar prin intermediul podului.

Podul orizontal trebuie să se afle la cel puțin un metru deasupra solului și la exact un metru sub cel mai de jos punct al cablului de suspensie.

De asemenea, podul trebuie să se afle la cel puțin doi metri sub cea mai de jos fereastră a celor doi.

Date de intrare

Fișierul de intrare **`suspense.in`** constă în mai multe teste. Fiecare test este descris prin trei linii.

Prima linie pentru un test va conține două numere întregi pozitive *j* și *t* și un număr real *d*, separate prin spațiu, care reprezintă etajul la care locuiește *Jan*, etajul la care locuiește *Tereza*, respectiv distanța dintre cele două clădiri exprimată în metri.

Cea de-a doua linie conține *j* litere mari ale alfabetului englezesc separate printr-un spațiu, iar cea de-a treia linie corespunzătoare unui test conține *t* litere mari ale alfabetului englez separate printr-un singur spațiu. Aceste litere reprezintă tipul de vietăți care trăiesc în etajele celor două clădiri, care sunt mai mici sau egale cu etajele la care locuiesc cei doi, și pot avea valorile `B` (la etajul respectiv se află un canar), `C` (la etajul respectiv se află o pisică) sau `N` (la etajul respectiv nu se află nici una dintre cele două tipuri de vietăți de mai sus). Vietățile din cele două clădiri sunt date pe linia a doua și a treia în ordinea crescătoare a etajelor de la 1 la *j*, respectiv de la 1 la *t*.

Ultimul test din fișierul de intrare este urmat de o singură linie care conține trei valori `0` separate între ele printr-un spațiu.

Date de ieșire

Pentru fiecare test din fișierul de intrare trebuie tipărit numărul acestuia (1, 2, ...) și cea mai mare valoare *c* cu proprietatea că două cabluri de lungime *c* pot fi folosite pentru a suspenda podul de capetele inferioare a ferestrelor celor doi elevi, astfel încât suprafața orizontală a podului să se afle la exact un metru sub cel mai de jos punct al celor două cabluri, la cel puțin un metru deasupra solului, la cel puțin doi metri sub cele două ferestre ale elevilor și nu permite nici unei pisici să ajungă la un canar.

Valoarea *c* trebuie rotunjită la trei zecimale exacte.

Dacă nu poate fi construit un pod care să respecte condiția din enunț trebuie afișat cuvântul `impossible`.

Pentru fiecare test din enunț se va tipări o linie de forma:

```
Case #: c
```

sau

```
Case #: impossible
```

unde `#` reprezintă numărul testului, iar *c* reprezintă valoarea calculată rotunjită la trei zecimale exacte.

Liniile furnizate la ieșirea standard trebuie despărțite printr-o linie vidă, ca în exemplu.

Restricții și precizări

- $2 \le j, t \le 25$;
- $1 \le d \le 25$;
- etajele sunt identificate prin numere întregi, iar cel mai de jos etaj este identificat prin 1;
- toate camerele din cele două clădiri au exact trei metri înălțime;
- toate ferestrele ale celor două clădiri au exact 1,5 metri înălțime și capătul de jos al fiecărei ferestre se află la exact un metru deasupra podelei fiecărei camere;

Exemplu

`suspense.in`	*Ieșirea standard*

`4 3 5.0`
`N C N C`
`N B B`
`4 3 5.0`
`C B C C`
`B C B`
`0 0 0`
	`Case 1: 14.377`
	
	`Case 2: impossible`

P040410: Controlul traficului aerian

Pentru a evita coliziunile din aer, majoritatea zborurilor comerciale sunt monitorizate de la sol de centre de control a traficului aerian care înregistrează pozițiile avioanelor folosindu-se de sisteme radar.

Datele pentru această problemă constau în două mulțimi: o mulțime de avioane și o mulțime de centre de control și trebuie să determinați cum trebuie distribuită monitorizarea avioanelor între centrele de control. Poziția fiecărui avion este descrisă de o pereche unică (x, y) de coordonate. Pentru scopul acestei probleme altitudinea la care se află avioanele poate fi ignorată.

Numărul de avioane care pot fi monitorizate de către un centru de control variază din când în când datorită schimbărilor de echipament și a angajaților.

La orice moment de timp, fiecare centru de control monitorizează atâtea avioane câte poate, în funcție de următoarele priorități:
- se preferă monitorizarea avioanelor care sunt mai apropiate de centrul de control;
- dacă două avioane sunt egal depărtate de centrul de control, și centrul poate să monitorizeze numai unul dintre

ele, atunci va fi monitorizat cel mai dinspre nord (valoare pozitivă pentru coordonata *y*);
- dacă două avioane sunt egal depărtate de centrul de control și au aceeași valoare pentru coordonata *y*, atunci este prioritar cel mai dinspre est (valoare pozitivă pentru coodonata *x*).

În orice moment, fiecare centru de control dispune de o arie circulară de control a cărei rază este dată de distanța dintre el și cel mai îndepărtat avion monitorizat. Toate avioanele cuprinse în această arie de control sunt monitorizate de centrul de control respectiv. Avioanele care se află chiar pe frontiera unui centru de control pot sau nu pot fi monitorizate de acesta, în funcție de prioritățile amintite mai sus.

Voi nu veți cunoaște pozițiile centrelor de control, în schimb, pentru fiecare centru de control veți cunoaște numărul de avioane pe care acesta îl monitorizează la un moment dat și coordonatele a două puncte care se află pe frontiera ariei de control. Folosind aceste informații, veți putea calcula poziția centrului de control și puteți decide care avioane sunt monitorizate de acesta.

Dacă datele sunt consistente în raport cu două sau mai multe arii de control, trebuie aleasă aria de control care include avionul cel mai nordic, iar dacă există două avioane având aceeași coordonată *y*, se va alege aria de control care monitorizează pe cel mai estic dintre avioane.

Figura de mai jos ilustrează patru avioane și două centre de control. Fiecare centru de control este reprezentat prin două puncte de pe frontieră, etichetate cu *A* și *B*, iar avioanele sunt etichetate cu *P*1, *P*2, *P*3 și *P*4. În acest exemplu avioanele *P*1 și *P*4 sunt monitorizate de un singur centru de control, avionul *P*3 este monitorizat de două centre de control, iar avionul *P*2 nu este monitorizat de nici un centru de control.

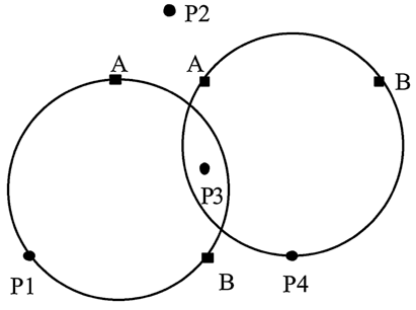

Figura 5

Date de intrare

Datele de intrare ce citesc din fișierul **traffic.in** și constau în mai multe teste. Datele pentru un test constau în mai multe linii.

Prima linie a unui anumit test conține două numere întregi *NP* și *NC*, separate printr-un singur spațiu, care reprezintă numărul de avioane, respectiv numărul centrelor de control. Fiecare dintre următoarele *NP* linii conține două numere reale *x* și *y* care reprezintă coordonatele la care se află un avion. Fiecare dintre următoarele *NC* linii conține cinci numere separate între ele prin spațiu. Primul număr este întreg, este cuprins între 0 și *NP* inclusiv și reprezintă numărul de avioane monitorizate de un centru de control; următoarele două numere sunt reale și reprezintă coordonatele primului punct aflat pe frontiera ariei de control a centrului respectiv, iar ultimele două numere sunt reale și reprezintă coordonatele celui de-al doilea punct aflat pe frontiera ariei de control a centrului.

Ultimul test este urmat de o linie pe care se află două valori 0 separate între ele printr-un singur spațiu.

Date de ieșire

Rezultatele vor fi furnizate la *ieșirea standard*.

Pentru fiecare test trebuie calculate numărul de avioane care nu sunt monitorizate de nici un centru de control, numărul de avioane care sunt monitorizate de un centru de control și așa mai departe până la numărul de avioane care sunt monitorizate de *NC* centre de control. Aceste valori vor trebui scrise pe o singură linie sub forma:

`Trial #:` n_0 n_1 n_2 ... n_{NC}

unde # reprezintă numărul testului, iar n_0, n_1, n_2, ..., n_{NC} reprezintă valorile calculate.

Dacă datele pentru un centru de control din cadrul unui anumit test sunt inconsistente, pentru testul respectiv se va scrie la ieșirea standard o linie de forma:

`Trial #: Impossible`

unde # reprezintă numărul testului.

Liniile furnizate la ieșirea standard trebuie despărțite printr-o linie vidă, ca în exemplu.

Restricții și precizări

- $0 < NP < 100$;
- $0 < NC < 10$;
- dacă diferența a două distanțe este mai mică decât 0,00001, atunci distanțele trebuie tratate ca fiind egale.

Exemplu

traffic.in

```
4 2
3.0 0.0
0.0 0.0
1.6 2.8
2.0 1.0
2 1.0 2.0 2.0 0.0
2 2.0 2.0 4.0 2.0
2 1
0.0 0.5
0.0 -0.5
0 -1.0 0.0 1.0 0.0
0 0
```

Ieșirea standard

```
Trial 1: 1 2 1

Trial 2: Impossible
```