The Tenth Hunan Collegiate Programming Contest

Online Version

(with one more problem, different input data and tighter time limits)

18th October, 2014
You get 15 Pages
12 Problems
&
300 Minutes

A

Arc and Point

Input: Standard Input
Output: Standard Output

Given a *circular* arc (i.e. part of the circumference of a circle) and a point P, your task is to calculate the shortest distance between them. That means, you should find a point on the arc, whose distance to P is minimized.

Attention: Try to use exact algorithms. Approximation algorithms are harder to pass the judge data.

Input

There will be at most 10000 test cases. Each case contains 8 integers x1, y1, x2, y2, x3, y3, xp, yp. The arc starts from A(x1,y1), goes through B(x2,y2) and ends at C(x3,y3). The point is located at (xp,yp). It is guaranteed that A, B, C are different points and will not be collinear. The absolute values of all coordinates are not greater than 20.

Output

For each test case, print the case number and the distance, to three decimal places. Absolute error of 0.001 is allowed.

Sample Input

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

Output for Sample Input

```
Case 1: 1.414
Case 2: 4.000
```

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan

B

Block Toy

Input: Standard Input
Output: Standard Output

There is a rectangular board, we want to build a toy by piling some unit blocks onto it. The toy can be described by the following "height matrix", which means we need 4 unit blocks in the middle, and 1 unit block at other positions.

1	1	1
1	4	1
1	1	1

We have an unlimited supply of 1x1 and 1x2 blocks, so we can build the toys in various ways. For example (letters are unit blocks, unit blocks with same letter belongs to the same 1x2 block):

```
AAB
DEB
DCC
```

(a) Top view

```
 E
 E
 F
DCC
```

(b) Front view

If at least one 1x1 blocks is used we say it's a silver toy, otherwise we say it's a gold toy.

Given the height matrix, find out the number of silver toys and gold toys we can build.

Input

There will be at most 20 test cases. Each test case begins with two positive integers R, C (1<=R*C<=16), the number of rows and columns. Each of the following R lines contains C integers h(i,j). (0<=h(i,j)<=20).

Output

For each test case, print the case number, the number of silver toys and the number of gold toys, both modulo 10^9+7.

Sample Input

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

Output for Sample Input

```
Case 1: 485 2
Case 2: 8 0
Case 3: 2794 12
```

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan, Shiplu Hawlader

C

Cool Word

Input: Standard Input
Output: Standard Output

A word is a string of lower-case letters. A cool word has at least 2 different letters and the number of occurrences of each different letter is different.

Here is a formal definition. Let *w* be a word and S be the set of letters in word *w*, then *w* is cool if and only if all f(c) (for each character c in S) is all different. Here f(c) means the number of occurrences of c in *w*.

For example, the word "ada" is cool because f(a)=2, f(d)=1, and they're different. "banana" is also cool because f(a)=3, f(n)=2, f(b)=1. But the word "bbacccd" is not cool because f(a)=f(d)=1. Some other interesting cool words include: mammal, needed, papaya, referee, senselessness.

Read a list of words and count the number of cool words.

Input

There will be at most 30 test cases. Each case begins with an integer n (1<=n<=10000), the number of words to check. Each of the following n lines contains a word containing at least one and at most 30 letters.

Output

For each test case, print the case number and the number of cool words.

Sample Input

```
2
ada
bbacccd
2
illness
a
```

Output for Sample Input

```
Case 1: 1
Case 2: 0
```

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan, Feng Chen

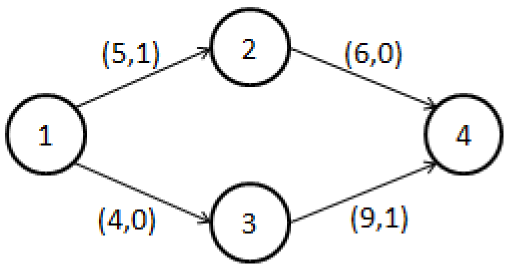
D

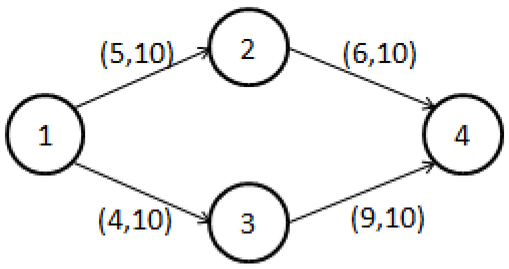
Double Shortest Paths

Input: Standard Input
Output: Standard Output

Alice and Bob are walking in an ancient maze with a lot of caves and one-way passages connecting them. They want to go from cave 1 to cave n. All the passages are difficult to pass. Passages are too small for two people to walk through simultaneously, and crossing a passage can make it even more difficult to pass for the next person. We define d_i as the difficulty of crossing passage i for the first time, and a_i as the additional difficulty for the second time (e.g. the second person's difficulty is d_i+a_i).

Your task is to find two (possibly identical) routes for Alice and Bob, so that their total difficulty is minimized.





For example, in figure 1, the best solution is 1->2->4 for both Alice and Bob, but in figure 2, it's better to use 1->2->4 for Alice and 1->3->4 for Bob. **It's always possible to reach cave n from cave 1.**

Input

There will be at most 200 test cases. Each case begins with two integers n, m (1<=n<=500, 1<=m<=2000), the number of caves and passages. Each of the following m lines contains four integers u, v, d_i and a_i (1<=u,v<=n, 1<=d_i<=1000, 0<=a_i<=1000). Note that there can be multiple passages connecting the same pair of caves, and even passages connecting a cave and itself.

Output

For each test case, print the case number and the minimal total difficulty.

Sample Input

```
4 4
1 2 5 1
2 4 6 0
1 3 4 0
3 4 9 1
4 4
1 2 5 10
2 4 6 10
1 3 4 10
3 4 9 10
```

Output for Sample Input

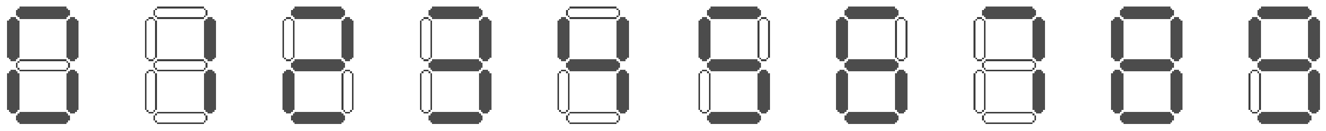
```
Case 1: 23
Case 2: 24
```

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan, Feng Chen

E Extraordinarily large LED

Input: Standard Input
Output: Standard Output

You're arranging a basket ball match in your university. In order to attract more people to watch the game, you decided to use an extraordinarily large LED display for the scores. However, the huge LED will consume a lot of power, so you would like to estimate the cost by analyzing some past games in your university.



Your LED display consists of 6 traditional seven-segment digits, 3 digits for each team. When a segment is on, it consumes one unit of power per second. When off, it does not consume any power at all. Scores are shown without leading zeros except that for score 0, a single digit "0" is shown.

Input

There will be at most 100 test cases. For each case, the first line is always "START hh:mm:ss" which means the game starts at hh:mm:ss. The last line is always "END hh:mm:ss" which means the game ends at the beginning of hh:mm:ss. There will be at least one SCORE information between START and END, formatted as "SCORE hh:mm:ss team score", where team is either "home" or "guest", score is 1, 2 or 3. The information will be sorted in increasing order of time, and no two times are equal. The game will start no earlier than 9:00 and will end no later than 21:00 in the same day. Note that if the start time is 09:00:00 and the end time is 09:00:01, the game duration is 1 second, not 2 seconds.

Output

For each test case, print the case number and the total power consumed.

Sample Input

```
START 09:00:00
SCORE 09:01:05 home 2
SCORE 09:10:07 guest 3
END 09:15:00
START 09:00:00
SCORE 10:00:00 home 1
SCORE 11:00:00 home 1
SCORE 12:00:00 home 1
SCORE 13:00:00 home 1
SCORE 14:00:00 home 1
SCORE 15:00:00 home 1
SCORE 16:00:00 home 1
SCORE 17:00:00 home 1
SCORE 18:00:00 home 1
SCORE 19:00:00 home 1
SCORE 20:00:00 home 1
END 21:00:00
```

Output for Sample Input

```
Case 1: 9672
Case 2: 478800
```

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan, Feng Chen

F

Four coloring of a map

Input: Standard Input
Output: Standard Output

You have a color map of an ancient kingdom. The map is a grid with R rows and C columns; **each region is a set of 4-connected cells**. Currently, adjacent regions are painted with different colors, but different regions may use the same color. You decided to re-color it with 4 colors: RED, GREEN, BLUE and YELLOW, according to the same rule (adjacent regions have different colors).

Coloring by yourself is tedious, so you invited your girlfriend to work with you. You paint with RED and GREEN, and your girlfriend paints with BLUE and YELLOW. You don't want your girlfriend to be tired, so you ask her not to paint more than 5 regions (painting exactly 5 regions is ok).

Your task is to count the number of different maps that could be painted. **Note that each color must be used at least once.**

Input

There will be at most 100 test cases. Each case begins with two integers R and C in the first line (1<=R,C<=20), the number of rows and columns in the map. Each of the next R lines contains C upper-case letters, the current color of each cell. **There will be at most 30 regions in each map, and most test cases have fewer regions.**

Output

For each test case, print the case number and the number of ways to color the map.

Sample Input

```
2 4
AABB
BBAA
1 5
ABABA
4 7
AABAABB
ABBCCCB
BBAACBB
CCABBAC
```

Output for Sample Input

```
Case 1: 24
Case 2: 144
Case 3: 3776
```

Problemsetter: Rujia Liu,Special Thanks: Md. Mahbubul Hasan, Feng Chen, Shiplu Hawlader

G

Giving directions to the tree

Input: Standard Input
Output: Standard Output

There is a rooted tree, some edges are undirected, while others are directed. We want to change maximum number of undirected edges to directed edges, but we don't want the length of the *longest* directed chain to be increased. Note that undirected edges are not allowed in a directed chain.

For example, if we have a "linear graph" 1->2->3-4, we cannot change 3-4 into 3->4 because the previous longest directed chain (1->2->3) would be extended to 1->2->3->4. However, we can change 3-4 into 3<-4 without extending the longest chain.

Input

There will be at most 1200 test cases. Each test case contains several lines. In each line, the first integer u is the node that is described, followed by its sons, terminated by a zero. The direction of an edge can be from father to son, and can also be from son to father. If the edge is from father to son, then we put a letter ``d" after that son (meaning that it is a downward edge). If the edge is from son to father, then we put a letter ``u" after that son (meaning that it is an upward edge). If the edge is undirected then we do not put any letter after the son. Nodes are numbered 1 to n (2<=n<=300) from top to down, left to right (so the first line is always root). Leaves are not given in the input. The test case ends with u=0. **Most test cases have very few nodes.**

Output

For each test case, print the case number, the number of changed edges, followed by the changed edges with directions. Each directed edge is formatted as (i,c), which means the i-th undirected edge is changed to direction c. The undirected edges are numbered from 1, in the same order they appear in the input.

If there are several optimal solutions, print the lexicographically smallest one. When comparing two changes (i1,c1) and (i2,c2) lexicographically, we first compare i1 and i2, if i1 and i2 are equal, we compare c1 and c2. For example (2,u) < (11,d), and (3,d) < (3,u).

Sample Input

```
1 2d 3 0
3 4 5 0
0
1 2d 0
2 3d 0
3 4 0
0
1 2d 0
2 3 0
3 4u 0
0
1 2u 3 0
3 4u 5 0
0
```

Output for Sample Input

```
Case 1: 3 (1,d) (2,u) (3,u)
Case 2: 1 (1,u)
Case 3: 0
Case 4: 1 (2,u)
```

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan, Shiplu Hawlader

H

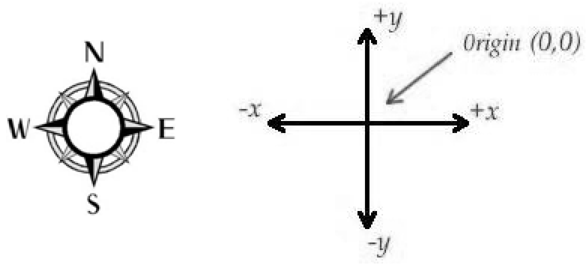
Happy Robot

Input: Standard Input
Output: Standard Output

A robot is moving from (0,0) according to a command sequence. Each character in the sequence is command:

- L: turn left
- R: turn right
- F: go forward one step

Interestingly, the command sequence contains some wildcard character "?". The robot can treat it any one of L, R or F at its own wish, which makes it really happy.



Let (x,y) be the final position of the robot, your task is to find out the minimal/maximal possible value of x and y. Initially the robot is facing east (i.e. facing (1,0) in Cartesian coordinate system). After a left turn it will face north (i.e. facing (0,1)).

Input

There will be at most 1000 test cases. Each case contains a command sequence with no more than 1000 characters.

Output

For each test case, print the case number, followed by minimal/maximal possible x (in this order), then the minimal/maximal possible y.

Sample Input

```
F?F
L??
LFFFRF
```

Output for Sample Input

```
Case 1: 1 3 -1 1
Case 2: -1 1 0 2
Case 3: 1 1 3 3
```

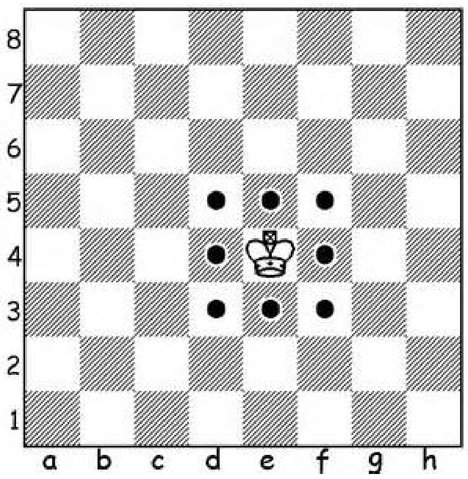
Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan, Feng Chen

I

Incomplete Chessboard

Input: Standard Input
Output: Standard Output

In chess, King is the most important piece. It can move left, right, up, down or diagonally, but only one square at a time, shown below.



Given two squares A(r1,c1), B(r2,c2), your task is to calculate the number of moves needed to move a king from A to B. To make the problem (slightly) harder, one square C(r3,c3) is removed from the chessboard, that means the king should never go into square C during his trip. In this problem, rows are numbered 1~8 from bottom to top, and columns are numbered 1~8 from left to right.

Input

There will be at most 10000 test cases. Each case contains 6 integers r1, c1, r2, c2, r3, c3 (1<=r1, c1, r2, c2, r3, c3<=8). Three squares A, B, C are always distinct.

Output

For each test case, print the case number and the minimum number of moves needed.

Sample Input

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

Output for Sample Input

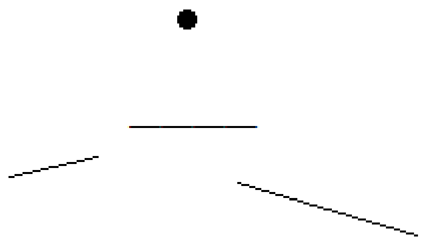
```
Case 1: 7
Case 2: 3
```

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan, Feng Chen

J	# Just another pachinko-like machine **Input:** Standard Input **Output:** Standard Output	

Like pachinko? Here is another one. It's not exactly a traditional pachinko, but it's also a let-the-ball-hit-things game.

In the machine, there are n non-overlapping non-vertical bars, shown below. Here non-overlapping means for every pair of bars, the two segments do not intersect, do not have common end-point, and do not partially overlap (they may overlap when projected to x-axis, though).



At the i-th step, the ball will be transferred to (x_i, y_i), then start to fall vertically, hopefully it'll hit a bar and earn some scores. A ball who hit the i-th bar will earn a score of s_i. If the ball directly drops on the floor (with y=0), it will not score.

The most interesting part of the machine is: **if the i-th bar is hit during this step, it will disappear at that moment and re-appear after d_i steps**. For example, if a bar with d_i=3 is hit in the 5-th step, then it'll be missing during step 6 and 7, and will re-appear in step 8.

Input

There will be at most 5 test cases. Each test case begins with one integer n ($1<=n<=10^5$), the number of bars. Each of the next lines contains 5 integers x1, y1, x2, y2, s, d ($0<=x1<x2<=10^9$, 1<=y1,y2<=200000, 1<=s<=1000, 1<=d<=5), describing one bar. No two bars can have any common point (i.e. no intersection, can’t touch each other etc).

The next line contains b ($1<=b<=10^5$), the number of balls. In the next b lines, the i-th line describes the ball appear in the i-th step. Each line contains two integers (x', y'), that means the ball will appear at (x_i,y_i)=(x' XOR a, y' XOR a), where a is the current score before the ball falls (which will be zero at the beginning of each test case). It is guaranteed that x_i and y_i are non-negative integers and will not be precisely on a bar.

Output

For each test case, print the case number in the first line and the scores after each step. There should be one empty line after each test case.

Sample Input

```
2
0 4 4 4 1 4
2 2 6 2 9 1
5
3 5
2 4
11 15
9 9
16 26
3
0 6 10 7 1 5
2 4 8 3 10 5
4 2 6 2 100 5
4
5 7
4 6
14 12
106 104
```

Output for Sample Input

```
Case 1:
1
10
10
19
20

Case 2:
1
11
111
111
```

Explanation for Sample 1

Step 1:
ball (3,5) will hit the first bar, score=1

Step 2:
ball (3,5) will hit the second bar, score=9

Step 3(bar 2 appear again):
ball (1,5) will hit the ground, score=0

Step 4:
ball (3,3) will hit the second bar, score=9

Step 5(bar 1&2 appear again):
ball (3,9) will hit the first bar again, score=1

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan

K	Kick the ball! **Input:** Standard Input **Output:** Standard Output	acm

Kick the ball!

Input: Standard Input
Output: Standard Output

"A penalty shoot-out (officially kicks from the penalty mark) is a method of determining the winner of an association football (soccer) match that is drawn after the regulation playing time and any applicable extra time periods have been played. In a penalty shoot-out, each team takes turns attempting a specified number of shots from the penalty mark (usually 5) that are only defended by the opposing team's goalkeeper, with the team scoring the most goals being declared the winner."

-- wikipedia

The game finally comes to the shoot-out. What will the final result be? "1-3!" You took a wild guess. But what is the probability that your guess is correct?

In this problem, team A kicks first (which is determined by a coin toss, as usual), both teams will attempt at most 5 shots (after all the 10 shots, the game may end in draw again), but the game will end as soon as the winner is already determined. For example, after the first 8 kicks the score is 3-2 (left side is team A's score, right side is team B), then if the 9-th kick is a goal, the game will end immediately with score 4-2, because even team B got its last kick, it still loses for sure. Another example: if all the first 9 kicks are goals, the last kick (from team B) will still be performed, because although team B cannot win, the result might be a "draw", which is better than "lose".

Input

There will be at most 100 test cases. Each case contains two lines. The first line contains 10 floating numbers. The first 5 numbers are the goal probability of the players in team A (player 1 will shoot first, etc), the next 5 numbers are the goal probabilities of the players in team B. Each probability will have exactly one digit after the decimal point. The second line contains your guess, in the format of scoreA-scoreB. 0<=scoreA,scoreB<=5.

Output

For each test case, print the case number and the probability (in percentage) that your wild guess is correct, to 2 decimal places. An absolute error of 0.01% will be ignored.

Sample Input

```
0.4 0.7 0.7 0.6 0.5 0.8 0.9 0.7 0.2 0.8
1-3
1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
2-0
1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0
2-0
0.4 0.7 0.7 0.6 0.5 0.8 0.9 0.7 0.2 0.8
5-5
0.4 0.7 0.7 0.6 0.5 0.8 0.9 0.7 0.2 0.8
4-2
```

Output for Sample Input

```
Case 1: 6.98%
Case 2: 100.00%
Case 3: 0.00%
Case 4: 0.47%
Case 5: 9.73%
```

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan, Feng Chen

L

Just another pachinko-like machine (II)

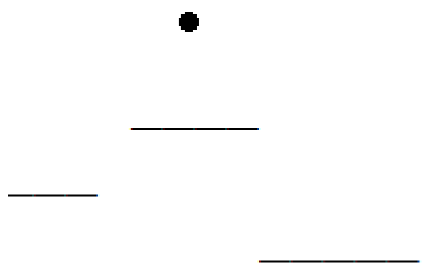
Input: Standard Input
Output: Standard Output

Summary:
There are only two different points between this problem and problem J:

- Bars are always horizontal.
- The input format for each bar is slightly different (because y1=y2).

Like pachinko? Here is another one. It's not exactly a traditional pachinko, but it's also a let-the-ball-hit-things game.

In the machine, there are n non-overlapping horizontal bars, shown below. Here non-overlapping means for every pair of bars, the two segments do not intersect, do not have common end-point, and do not partially overlap (they may overlap when projected to x-axis, though).



At the i-th step, the ball will be transferred to (x_i, y_i), then start to fall vertically, hopefully it'll hit a bar and earn some scores. A ball who hit the i-th bar will earn a score of s_i. If the ball directly drops on the floor (with y=0), it will not score.

The most interesting part of the machine is: **if the i-th bar is hit during this step, it will disappear at that moment and re-appear after d_i steps**. For example, if a bar with d_i=3 is hit in the 5-th step, then it'll be missing during step 6 and 7, and will re-appear in step 8.

Input

There will be at most 5 test cases. Each test case begins with one integer n ($1<=n<=10^5$), the number of bars. Each of the next lines contains 5 integers x1, x2, y, s, d ($0<=x1<x2<=10^9$, 2<=y<=200000, 1<=s<=1000, 1<=d<=5), describing one bar. No two bars can have any common point (i.e. no intersection, can't touch each other etc).

The next line contains b ($1<=b<=10^5$), the number of balls. In the next b lines, the i-th line describes the ball appear in the i-th step. Each line contains two integers (x', y'), that means the ball will appear at (x_i,y_i)=(x' XOR a, y' XOR a), where a is the current score before the ball falls (which will be zero at the beginning of each test case). It is guaranteed that x_i and y_i are non-negative integers and will not be precisely on a bar.

Output

For each test case, print the case number in the first line and the scores after each step. There should be one empty line after each test case.

Sample Input

```
2
0 4 4 1 4
2 6 2 9 1
5
3 5
2 4
11 15
9 9
16 26
3
0 10 6 1 5
2 8 4 10 5
4 6 2 100 5
4
5 7
4 6
14 12
106 104
```

Output for Sample Input

```
Case 1:
1
10
10
19
20

Case 2:
1
11
111
111
```

Explanation for Sample 1

Step 1:
ball (3,5) will hit the first bar, score=1

Step 2:
ball (3,5) will hit the second bar, score=9

Step 3(bar 2 appear again):
ball (1,5) will hit the ground, score=0

Step 4:
ball (3,3) will hit the second bar, score=9

Step 5(bar 1&2 appear again):
ball (3,9) will hit the first bar again, score=1

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan