IDI Open
Programming Contest
April 25th, 2009

The Problem set

Jury and Problem Writers

Eirik Reksten, IDI/NTNU
Børge Nordli, Google
Nils Grimsmo, IDI/NTNU
Truls A. Bjørklund, IDI/NTNU
Rune Fevang, Opera Software
Rune Johan Hovland, IDI/NTNU
Erling Alf Ellingsen
Erik Axel Nielsen, McKinsey & Company
Øyvind Grotmol, Medallia

Tips

- Tear the problem set apart and share the problems among you.
- Problems are not ordered by difficulty.
- Try solving the easy problems first. Two problems in this set are tagged with “(Easy)” to help point you in the right direction.
- If your solution fails on a problem, you can print your program and debug it on paper while you let someone else work on a different problem on the computer.
- If you need help, contact the judges.

Rules

- Each team consists of one to three contestants.
- One computer is used per team.
- You may not cooperate with persons not on your team.
- You may print your programs on paper to debug them.
- What you may bring to the contest:
  - Any written material (Books, manuals, handwritten notes, printed notes, etc).
  - Pens, pencils, blank paper, stapler and other useful non-electronic office equipment.
  - NO material in electronic form (CDs, USB pen and so on).
  - NO electronic devices (PDAs and so on).
- The only electronic content you may consult during the content is that specified by the organiser (see the web-page). You may not copy source code from web pages, etc.
- Your programs should read from standard in and write to standard out. Writing to standard error will result in a failed submission. C programs should return 0 from `main()`.
- Your program may use at most 100MB of memory.
- Your programs may not:
  - access the network,
  - read or write files on the system,
  - talk to other processes,
  - fork,
  - or similar stuff.
  - If you try, your program will hang or crash. If it hangs, it will take a couple of minutes before others will be able to run their programs. And please do not crack somebody who uses their spare time trying to give you something valuable.
- Show common sense and good sportsmanship.

Problem A

Letter Cookies

Sætre has finally decided to put their famous Letter Cookies back on market again. Of course they are just as interesting to play with as to eat. Your little sister is trying to make words out of the letters she found in the box, but you want to be faster than her to decide whether it is possible to make the word or not.

Given the letters in the cookie box, is it possible to spell out the words your little sister knows how to spell? (After creating a word, she scrambles the cookies again and can reuse the letters for later words.)

Input specifications

The first line of the input consists of a single number T, the number of letter cookie boxes your sister has. Each test case starts with a line describing all the letters in this box, in no particular order. Then follows a line with W, the number of words she would like to spell, and then follow the W words to write on a single line each.

Output specifications

For each word, output a line containing `YES` if it is possible to spell the word, or `NO` if it is not possible.

Notes and Constraints

- $0 < T \leq 100$
- $0 < W \leq 100$
- All letters are uppercase letters from the english alphabet (`A-Z`).
- There are at most 1000 letters in each cookie box.
- Each word has at most 100 letters (but is not necessarily an actual English word).

Sample input

```
1
ABANANACOOKIE
4
BANANA
LETTER
COOKIES
CAN
```

Output for sample input

```
YES
NO
NO
YES
```

Problem B

Bicycle puzzle

Per and Gunnar has found a wonderful online bicycle picture puzzle flash solitaire game, and since they are very competitive, they want to find out who is best. The purpose of the game is to descramble a picture of a bicycle. At the start of each game, the picture of the bicycle is cut into W times H equally sized rectangles and scrambled randomly. Then the player repeatedly chooses two arbitrary rectangles and swaps them. He continues to swap rectangles until the picture is complete again. The game keeps track of the number of swaps, which is the score for that particular playthrough.

After Gunnar has played a game, he sends his score (along with W and H) to Per, and challenges Per to beat this score. Per soon realizes that if he is unlucky with the scrambling, it is impossible to beat Gunnar's score.

Per quickly writes a program to find out the probability that he can beat Gunnar's score (assuming that all scrambled pictures are equally probable) if he plays optimally. But he is not sure whether his program is correct and wants to verify its correctness by challenging you to write the same program.

Input specifications

The first line of the input consists of a single number T, the number of scenarios. Each scenario is given by a line with three integers W, H and S, where S is Gunnar's last score.

Output specifications

For each scenario, output a line with the probability that Per beats Gunnar's score. Format the probability as an irreducible fraction where the numerator and the denominator are separated by `/`. Output only the numerator if the answer is an integer.

Notes and Constraints

- $0 < T \leq 150$
- $0 < W \leq 5$
- $0 < H \leq 4$
- $0 \leq S \leq W \cdot H$
- When comparing two scores, the lowest one is best.

Sample input

```
3
1 1 1
1 2 1
3 1 2
```

Output for sample input

```

1
1/2
2/3
```

Problem C

Geometry Darts

Bob and Hannah like to play darts. They are not very good at it, however, so finishing a round of 501 Darts will take an eternity. They therefore decide to throw away the dartboard completely and put geometric shapes on the wall instead, awarding points according to the number of shapes the dart penetrates. To reduce the complexity of scoring, they only use circles, triangles and rectangles.

A game consists of each person throwing 3 darts each, and your job is to find the winner of the game, given the shapes and the throws.

Input specifications

The input will start with a line giving the total number of shapes, S. Then follow S lines describing the shapes, in either of the following formats:

1. `C` x y r, where (x, y) is the center of the circle, and r is the radius.
2. `R` x_1 y_1 x_2 y_2, where (x_1, y_1) and (x_2, y_2) are two corners of the rectangle with $x_1 < x_2$ and $y_1 < y_2$.
3. `T` x_1 y_1 x_2 y_2 x_3 y_3, where (x_i, y_i) are the three corners of the triangle.

Then follows a line with N, the number of games Bob and Hannah play. Each game is described with six lines giving the x and y coordinates of the 6 throws, the first three by Bob and the last three by Hannah.

Output specifications

Output the name of the winner on a separate line for each game, or `Tied` if there is a tie.

Notes and Constraints

- $0 < S \leq 1000$
- $0 < N \leq 1000$
- All rectangles have sides parallell to the x- and y-axis.
- For triangle specifications, the three points will never be collinear.
- All coordinates are given with double precision, with up to 6 decimals after the decimal points.
- All shapes are bounded by the rectangle defined by the two points $(-1000, -1000)$ and $(1000, 1000)$.
- All throws are guaranteed to be at least 10^{-6} away from any shape boundary.

Sample input

```
3
C 0.0 0.0 5.0
R -1.0 -1.0 7.0 7.0
T 0.0 0.0 -3.0 0.0 0.0 -8.0
1
0.0 4.1
0.0 6.2
0.0 8.1
-0.5 -0.5
-0.5 -2.0
-0.5 -5.1
```

Output for sample input

```
Hannah
```

Problem D

Box Betting

Theres a new delivery boy at your company, supposed to drive around delivering a specific number of items to different locations. He needs to deliver at least X of an item, while the truck can at most take Y of the item. If there are more, the truck is will to break down due to the weight. If there are less, that means he hasn't fulfilled the assignment. At the loading station, there's a row of boxes he can bring along. Each box contains a specified number of items.

This driver isn't the sharpest snail on the rock, however. Since he's also too shy to ask for help, he's decided to just choose one entirely random starting point in the row of boxes. Then he chooses a random end point among the boxes from the starting point to the end, inclusive. All the boxes between those points (inclusive) are loaded onto the truck.

You and the other employees have started betting on whether or not the truck breaks down, he brings to few items along or if he should happen to be lucky enough to fulfil the assignment without any problems. This gets you wondering. What is the actual probability for each of these scenarios to occur? Assume for the sake of this problem that the driver will always be able to fit all the boxes into the truck (after all, he had to have SOME skill, seeing as he was hired).

Input specifications

The input will start with a line containing a single number T, the number of test cases. Each test case consists of three lines. The first one contains a single number N, the number of boxes. The second line contains a sequence of N characters (`A-Z`), $B_1B_2...B_N$ (no whitespace), representing the amount of items in each of the boxes in the same order as they're located on the loading dock. An `A` represents an empty box, `B` a box with 1 item, and so on until Z, which represents a box with 25 items in it. The third line contains the two numbers L and U. L is the number of items the driver is supposed to deliver, while U is the maximum number of items the truck can take before it'll break down.

Output specifications

For each test case, output three floating point numbers on a single line. The first number gives the probability that the driver succeeds, the second one that that he brings along too few items, and the third one that the truck breaks down.

Notes and Constraints

- $0 < T \leq 100$
- $1 < B \leq 200000$
- 'A' $\leq A_i \leq$ 'Z'
- $0 \leq L \leq U \leq 50000$
- Both the starting and ending points of the segment are chosen with a uniform probability distribution.
- Any answer within 10^{-6} of the correct one will be accepted.

Sample input

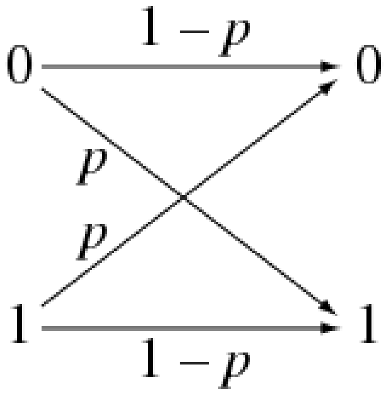
```
2
4
KCHA
2 9
3
BCD
4 5
```

Output for sample input

```
0.5 0.25 0.25
0.16666667 0.72222222 0.11111111
```

Problem E

Communication Channels (Easy)



Classical information theory is based on the concept of a communication channel.

> *Information theory is generally considered to have been founded in 1948 by Claude Shannon in his seminal work, "A Mathematical Theory of Communication." The central paradigm of classical information theory is the engineering problem of the transmission of information over a noisy channel.*
> http://en.wikipedia.org/wiki/Information_theory

In this problem, we will specifically consider one of the simplest possible noisy channels, namely the binary symmetric channel (BSC). A BSC transmits a sequence of bits, but each transmitted bit has a probability p of being flipped to the wrong bit. This is called the crossover probability, as can be understood from the figure. We assume independent behaviour on different bits, so a communication of l bits has probability $(1-p)^l$ of being transmitted correctly. Note that one can always assume that $p < 1/2$, since a channel with $p = 1/2$ is totally useless, and a channel with $p > 1/2$ can easily be transformed to a new channel having crossover probability $1-p$ by just flipping all bits of the output.

Of course, it is still possible to communicate over a noisy channel. (In fact, you are doing it all the time!) To be able to do this, one has to add extra bits in order for the receiver to detect or even possibly correct errors. Example implementations of such a feature are parity bits, Cyclic Redundancy Checks (CRC) and Golay codes. These are not relevant to this problem, however, so they will not be discussed here.

In this problem you must investigate the behaviour of a binary symmetric channel.

Input specifications

The first line of the input consists of a single number T, the number of transmissions. Then follow T lines with the input and the output of each transmission as binary strings, separated by a single space.

Output specifications

For each transmission, output `OK` if the communication was transmitted correctly, or `ERROR` if it was transmitted incorrectly.

Notes and Constraints

- $0 < T \leq 100$
- All inputs and outputs has length less than 120.
- T is encoded in decimal.

Sample input

```
2
10 10
10 11
```

Output for sample input

```

OK
ERROR
```

Problem F

Train Tickets

Tor Gunnar is a huge fan of trains. All his life, he's been running around at home yelling "Toot Toot". So when he recently found out about a job opening at a local train company, he immediately applied. Tor Gunnar, having studied really hard to reach such opportunities, got the job. It was the happiest moment of his life! To his great dissappointment, though, he discovered that the contract said nothing about riding trains all day. He is now assigned to the IT department, and is supposed to help write a new ticket management system. The part he is supposed to write, is an algorithm for distributing tickets among the different possible travels along a train route.

Tor Gunnar comes from a strange country. In addition to speaking a language even he himself doesn't understand, the countrys government has pushed through some weird laws that influences how train tickets are sold. This means that the price for travelling from station A to station B is predetermined. Also, everytime you sell tickets for a given trip, you'll know the exact demand for travelling between each pair of cities. What more, the government is allowed to set aside some tickets for their own usage (these are free of charge).

One train trip consists of a series of stations. It is possible to travel from station A to B as long as A precedes B in the station list. Each train has a given capacity, and there can never be more than this amount of passengers in between two adjacent stations.

Tor Gunnar is completely puzzled, and desperately needs your help. Write a program that helps him figure out the best ticket distribution.

Input specifications

The first line of input contains a single number T, the number of test cases to follow. Each test case begins with a line containing two numbers, N and P, the number of stations in the case and the capacity of the train, respectively. Then follow $N-1$ lines, giving the ticket prices, C_{ij}. The i-th of these N lines contain $N-i$ numbers. The j-th number of the i-th line is the cost of a ticket from station i to station $i+j$. The next $N-1$ lines give the demand for tickets for each pair of stations, D_{ij}, in the same format as the prices. Another $N-1$ lines in the same format follow these, statin the number of tickets set aside for the government officials, O_{ij}.

Output specifications

For each test case, output a single number on a line by itself. The number should equal the maximum possible income from the train trip.

Notes and Constraints

- $0 < T \leq 100$
- $3 \leq N \leq 16$
- $0 < P \leq 200$
- $0 < C_{ij} \leq 1000$
- $0 \leq D_{ij} \leq 250$
- $0 \leq O_{ij} \leq 20$
- A train trip goes from station 1 to station N.
- Keep in mind that the passenger count including the government officials cannot be more than the capacity.
- You are not allowed to overbook the train.
- The government will never overbook your train using their "free" tickets.

Sample input

```
1
3 4
6 7
3
4 1
1
2 1
0
```

Output for sample input

```
10
```

Problem G

Counting Sheep (Easy)

A while ago I had trouble sleeping. I used to lie awake, staring at the ceiling, for hours and hours. Then one day my grandmother suggested I tried counting sheep after I'd gone to bed. As always when my grandmother suggests things, I decided to try it out. The only problem was, there were no sheep around to be counted when I went to bed.

Creative as I am, that wasn't going to stop me. I sat down and wrote a computer program that made a grid of characters, where `#` represents a sheep, while `.` is grass (or whatever you like, just not sheep). To make the counting a little more interesting, I also decided I wanted to count flocks of sheep instead of single sheep. Two sheep are in the same flock if they share a common side (up, down, right or left). Also, if sheep A is in the same flock as sheep B, and sheep B is in the same flock as sheep C, then sheeps A and C are in the same flock.

Now, I've got a new problem. Though counting these sheep actually helps me fall asleep, I find that it is extremely boring. To solve this, I've decided I need another computer program that does the counting for me. Then I'll be able to just start both these programs before I go to bed, and I'll sleep tight until the morning without any disturbances. I need you to write this program for me.

Input specifications

The first line of input contains a single number T, the number of test cases to follow. Each test case begins with a line containing two numbers, H and W, the height and width of the sheep grid. Then follows H lines, each containing W characters (either `#` or `.`), describing that part of the grid.

Output specifications

For each test case, output a line containing a single number, the amount of sheep flocks on that grid according to the rules stated in the problem description.

Notes and Constraints

- $0 < T \leq 100$
- $0 < H, W \leq 100$

Sample input

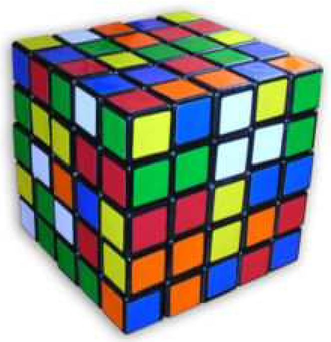
```
2
4 4
#.#.
.#.#
#.##
.#.#
3 5
###.#
..#..
#.###
```

Output for sample input

```
6
3
```

Problem H

Rubiks Cube



Following the hype generated by Rubiks Cube Norwegian Open Championship arranged at NTNU this year, each and every student at NTNU has bought such a cube. The professors, however, are dismayed, because the students rather play with their cube instead of listening to the professors lectures.

One professor suddenly gets the idea that if he gives the students a program to solve their cubes they may lose interest in it, so that it will be possible to start teaching again. Of course, the professor don't want to do the grunt work of programming, and you are the lucky assignee of this task. The professor takes away your cube, locks you in a faraway laboratory and says that you won't get out until you have written a program to solve Rubiks cube.

Luckily, the professor did not specify the size of the cube, so you decide to make the work slightly easier by solving the Rubiks $2 \times 2 \times 2$ cube.

Input specifications

The first line of the input consists of a single number T, the number of test cases. Each test case consists of six lines describing the initial configuration of a cube, formatted exactly as in the example input. The characters used for colors are `G`, `R`, `O`, `B`, `Y` and `W`. Each test case is followed by an empty line.

Output specifications

For each scenario, output a line with the minimum number of moves that is necessecary to solve the cube. A move is turning one face of the cube 90 degrees clockwise or counter-clockwise. (A turn of 180 degrees is considered to be two moves.)

Notes and Constraints

- $0 < T \leq 100$
- You may assume that all input cubes are scrambled versions of originally solved cubes with six different colors. (Which means there are exactly four occurrences of each color and that it is possible to get the cube to a solved state using moves as described above.)

Sample input

```
2
OO
OO
RR GG BB WW
RR GG BB WW
YY
YY

RR
RR
YY OO GG BB
OO GG BB YY
WW
WW
```

Output for sample input

```
0
1
```

Problem I

Marble Madness

Øyvind likes to test his future employees, especially with funny games. This time he has invented a game where you start out with B black marbles and W white ones in a bag. You also have endless supplies of both kinds of marbles outside the bag. The game proceeds in rounds. In each round, you take two marbles at random out of the bag, and put one marble back in (possibly another color than any of the two you took out), obeying the following rules:

1. When you take out two white marbles, you put a black one back in.
2. When you take out one black and one white, you put a white marble back in.
3. When you take out two black marbles, you put a black one back in.

At the end of this game, there will only be one marble in the bag. This marble may be white with a probability and black with a probability given the number of marbles of each colour you start with. Your job is to find these probabilities.

Input specifications

The input will start with a line giving the number of test cases, T. Each test case will be presented at one line with two integers, B and W, separated by a single space, representing the number of black and white marbles respectively.

Output specifications

There should be one line of output for each test case with two floating point numbers separated by a single space. The first number should represent the probability that the last marble is black, and the second the probability that the last marble is white.

Notes and Constraints

- $0 < T \leq 100$
- $0 \leq B, W \leq 50000$
- $0 < B + W$
- Any answer within 10^{-6} of the correct one will be accepted.

Sample input

```
2
1 1
3 0
```

Output for sample input

```

0.00 1.00
1.00 0.00
```

Problem J

Robberies

The aspiring Roy the Robber has seen a lot of American movies, and knows that the bad guys usually gets caught in the end, often because they become too greedy. He has decided to work in the lucrative business of bank robbery only for a short while, before retiring to a comfortable job at a university.

For a few months now, Roy has been assessing the security of various banks and the amount of cash they hold. He wants to make a calculated risk, and grab as much money as possible. His mother, Ola, has decided upon a tolerable probability of getting caught. She feels that he is safe enough if the banks he robs together give a probability less than this.

Input specifications

The first line of input gives T, the number of cases. For each scenario, the first line of input gives a floating point number P, the probability Roy needs to be below, and an integer N, the number of banks he has plans for. Then follow N lines, where line j gives an integer M_j and a floating point number P_j. Bank j contains M_j millions, and the probability of getting caught from robbing it is P_j.

Output specifications

For each test case, output a line with the maximum number of millions he can expect to get while the probability of getting caught is less than the limit set.

Notes and Constraints

- $0 < T \leq 100$
- $0.0 \leq P \leq 1.0$
- $0 < N \leq 100$
- $0 < M_j \leq 100$
- $0.0 \leq P_j \leq 1.0$
- A bank goes bankrupt if it is robbed, and you may assume that all probabilities are independent as the police have very low funds.

Sample input

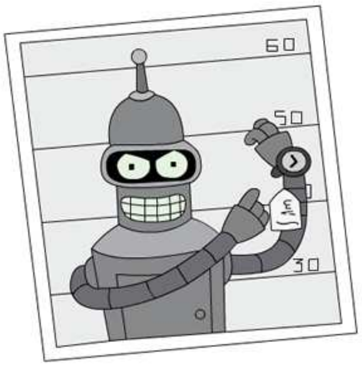
```
3
0.04 3
1 0.02
2 0.03
3 0.05
0.06 3
2 0.03
2 0.03
3 0.05
0.10 3
1 0.03
2 0.02
3 0.05
```

Output for sample input

```
2
4
6
```

Problem K

Robotic Encryption



Due to suspicion of cheaters, one of the more paranoid problem setters has started encrypting all messages to the rest of the jury before sending them. He didn't use any standard encryption, however, as he believes those are all part of a giant conspiracy network trying to crush IDI Open from the inside. Instead, he based it on the fact that the cheaters are likely to be the worst programmers. The decryption requires some programming skill, and should therefore be safe.

Along with the encrypted message, he sent explanation of how to decrypt it. The only problem now is that not all jury members are able to implement the decryption. This is where we need your help. You need to help us decrypt these messages by writing a program that does the task.

Decryption is performed by simulating a robots movement on a grid. The robot is initially placed in the north-west corner of the grid, facing south. The robot is a simple one, and only accepts three different commands:

- `L` turns the robot 90° to the left.
- `R` turns the robot 90° to the right.
- `F` moves the robot one square forward. If moving forward would cause the robot to fall off the grid, the robot instead makes a 180° turn without moving.

Instructions to the robot is given in a series of commandsets. A commandset is a string of commands, with the possible addition of loops. A loop is given on the form "(commandset)number" where number is the number of times the commandset inside the parentheses should be run. Longer sequences of commands can be built up recursively in this fashion. More formally:

commandset ::= *instruction+*
instruction ::= *command*|*loop*
loop ::= "(" *commandset* ")" *number*
command ::= `R` | `L` | `F`
number ::= `1` | `2` | `3` | `4` | `5` | `6` | `7` | `8` | `9`

The decrypted text is the text string obtained by concatenating the characters on the grid positions the robot stands on after executing each commandline.

Input specifications

The first line of input gives T, the number of test scenarios. Each scenario starts with a line containing W and H, separated by a single space, describing the dimensions of the grid. Then follows H lines, each consisting of W characters, making up the grid. After this comes a line containing N, the number of commandlines, followed by the N lines the robot will be executing.

Output specifications

One line per test scenario, containing the decrypted text.

Notes and Constraints

- $0 < T \leq 100$
- $0 < W \leq 50$
- $0 < H \leq 50$
- $0 < N \leq 20$
- Commandlines will be no longer than 50 characters, and will follow the syntax given in the problem text.
- No character with ASCII value lower than 32 or higher than 126 will appear on the robots grid.

Sample input

```
1
6 7
012345
6789AB
CDEFGH
IJKLMN
OPQRST
UVWXYZ
_! .,&
12
FFL(F)5
(F)4
(LF)2
(L(R)6L)9
RFRFFF
(L(F)2)2
LF
FLFF
FFFF
LF
FLFF
L(F)4
```

Output for sample input

```
HELLO WORLD!
```