IDI Open Programming Contest April 20th, 2013

Problem Set

Jury and Problem Writers

Christian Neverdal Jonassen, NTNU
Børge Nordli, Microsoft
Eirik Reksten, Steria
Ruben Spaans, NTNU
Jon Marius Venstad, NTNU

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. So please make an effort to not not crack/break what we have spent our spare time preparing for you.
- Show common sense and good sportsmanship.

Problem A

Angry Grammar Nazi

Your friend is what we can call a grammar nazi. He spends a lot of time on popular internet discussion forums. Unfortunately, he has a bad temper and loses his mind whenever someone incorrigibly befouls the English language, with unrelenting violations of grammatical and ortographic rules.

In order to avoid smashed keyboards, monitors and coffee-cup holders, you advice your friend to momentarily stop reading and count to ten each time he becomes angry, instead of smashing something.

Your friend becomes angry whenever he reads the following words or sequences of words:

- "u", "ur" instead or "you", "your".
- "would of", "should of" instead of "would have", "should have".
- "lol" instead of "haha". In fact he becomes angry even when a word contains "lol" as a substring.

You decide to write a computer program that reads sentences one by one, and for each sentence calculates how many times your friend will have uttered a number after reading said sentence. Your friend does not read out loud, so numbers that are part of the input-sentences should not be counted.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. The following T lines each contain one sentence; that is, one or more words separated by space.

Output specifications

For each test case, output how many times your friend have said a number after reading the sentence.

Notes and Constraints

- $0 < T \leq 50$
- A sentence consists of at most 100 characters, including spaces.
- A word consists only of lower case letters between `a` and `z`, inclusively.
- Two adjacent words are separated by exactly one space, and a sentence never has leading or trailing spaces.

Sample input

```
4
u haz lolcats
my car is green
i have a lollipop
u should of lold
```

Output for sample input

```

20
0
10
30
```

Problem B

Neurotic Network

In this problem, a neural net is represented by a rooted tree with weighted edges. The neural net processes information by a cascade of signals that begins at the leaf nodes: Each node in the tree computes an output value based on its upstream neighbors, and passes this value on to its downstream neighbor. The output value computed by a node is the sum of the output of each of its upstream neighbors multiplied by the weight of the edge from the upstream neighbor to the node itself. A node with no upstream neighbors (leaf nodes) always has 1 as output. All neural nets in this problem have exactly one final output node (the root node).

Sometimes, a neural net can go haywire and become what is more commonly known as a *neurotic* network. Consider this your chance to launch a second career in psychiatry. The scenario is that someone just came in with a neurotic network in their head. What this means is that if the output of their neural net is an even number, the person will freak out and will set fire to a kitten. Therefore, it is of vital importance that you can know ahead of time whether or not a given person is safe. If it is safe, print their neural output modulo 1,000,000,007. If you wouldn't trust the person to be around kittens who're not wrapped in fire retardant, print the string "`FREAK OUT`" (without the quotes).

Input specifications

The first line of input is T, the number of test cases. For each of the T cases, the first line will be the integer N, the number of nodes in the tree. The next line contains $N-1$ integers $a_1, a_2, \cdots, a_{N-1}$ where a_i is the downstream neighbour of the node with ID i. Then follows a line with $N-1$ integers $w_1, w_2, \cdots, w_{N-1}$ where w_i is the weight of the neural connection going out from the node with ID i. Note that the node with ID 0 will always be the output node.

Output specifications

Output "`FREAK OUT`" (without the quotes) if the final value of the neural net's output node is even. Otherwise, output the final value of the output node, modulo 1,000,000,007.

Notes and Constraints

- $0 < T \leq 50$
- $0 < N \leq 10000$
- $0 < w_i \leq 10$
- $0 \leq a_i < N$
- The graph is guaranteed to be a tree.
- This is an I/O-heavy problem. For Java programmers, this means that you should use `BufferedReader` for input reading (not `Scanner`).

Sample input

```
4
1


4
0 0 2
7 7 1
4
0 0 2
6 7 2
11
0 1 2 3 4 5 6 7 8 9
9 9 9 9 9 9 9 9 9 3
```

Output for sample input

```
1
FREAK OUT
FREAK OUT
162261460
```

Problem C

Special Services

You're assigned the job of making a booking system for a special services company. This company conducts all sorts of services, but you are told not to ask any questions. There have been some developers before you that have tried making this system, but sadly none of them are anywhere to be found, so you can't ask them for any help.

The system is receiving bookings (and cancellations) one by one, and must immediately accept/reject the booking. All bookings last for a whole day, and the part of the system you're making needs to keep track of a single day.

The company has a number of employees, each having one or more qualifications. Each booking, in addition to having been a given numeric identifier, demands a number of people (employees), each having a specific qualification. One person can cover at most one demand from one booking during the whole day. The director used Mario as an example; while Mario can be both a plumber and an assassin, in a single day he can only do one of the jobs for one booking (in this company).

A booking is accepted if and only if

- there are no active bookings with that specific identifier, *and*
- it is possible, using the given employees, to cover all the demands of the currently active bookings as well as the new one.

A cancellation is accepted if and only if

- there is an active booking with that specific identifier.

Input specifications

The first line contains T, the number of test cases that follow. Each test case starts with a line containing the numbers N and B, the number of employees and bookings/cancellations in the test case.

Then there follow N lines, each specifying an employee. Each line consists of a number, c, followed by c strings delimited by whitespace, giving the names of each qualification of that particular employee.

Subsequently there follows B lines, each containing either a booking or a cancellation. A booking starts with the word `book`. Then follows the numeric identifier I of that booking, followed by the number d and then finally d strings delimited by whitespace. This gives all qualification demands this booking has. A cancellation consists of a the word `cancel` followed by a number referring to the identifier I of the booking to be cancelled.

Output specifications

For each booking/cancellation, output a single line of output. The line should consist of the word `Accepted` if the booking/cancellation is accepted, or `Rejected` otherwise.

Notes and Constraints

- $0 < T \leq 50$
- $1 \leq N \leq 200$
- $1 \leq B \leq 1000$
- $1 \leq c \leq 16$
- $0 \leq d \leq 16$
- $0 \leq I \leq 1000$
- Qualification names will be at most 10 characters long. There will be no more than 100 different qualifications shared among all the employees.
- An active booking is one that has been accepted, and not successfully cancelled.
- It is possible to make bookings reserving no people whatsoever. These bookings will be handled by the company manager, and you should keep on asking no questions.
- This is an I/O-heavy problem. For Java programmers, this means that you should use `BufferedReader` for input reading (not `Scanner`).

Sample input

```
2
1 1
1 slacker
book 6 1 worker
3 6
2 plumber nurse
2 nurse assassin
2 assassin plumber
cancel 1
book 1 2 plumber assassin
book 123 1 nurse
book 4 3 assassin nurse plumber
cancel 1
book 4 2 nurse assassin
```

Output for sample input

```
Rejected
Rejected
Accepted
Accepted
Rejected
Accepted
Accepted
```

Problem D

Negative People in Da House (Easy)

The following math joke is presented for your amusement: Two mathematicians sit in a car outside a house. Two people enter the house. Then, three people are observed going out of the house. One of the mathematicians exclaim: If one person is to enter now, the house will be empty!

Since you have very little sense of humor, you are to write a program that will calculate the minimum number of people there must have been there to begin with. In other words, given a sequence of groups of people leaving and entering the house, output the minimum number of people there must have been **before you started stalking**. After writing this program, your mathematician friend will leave you, as well as their math department, to start a company specializing in joke-telling and stalking.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each of the following T cases has two parts: First, a line containing a single integer M. Then follows M lines with two integers P_1 and P_2 separated by a space, where the first one contains the number of people entering the house, then the number of people leaving the house. Note that these are two events: *First*, P_1 people enter the house, *then* P_2 people leave the house.

Output specifications

Output the minimum number of people that would have to have been inside the house at the beginning.

Notes and Constraints

- $0 < T \leq 50$
- $0 < M \leq 100$
- $0 \leq P_1, P_2 \leq 1000$

Sample input

```
1
3
3 5
3 4
1 0
```

Output for sample input

```
3
```

Problem E

Ruben Spawns (Easy)

Being the head judge of a programming contest is no small thing. As with everything good in life, there is that inevitably long list of things that need to be done before it can happen. Fortunately for Ruben, he has recently acquired a machine that at the push of a button, can spawn a small minion for him to do part of his work. He has also hired an assistant to remind him to activate the machine.

There is one caveat with minions. If you have too many of them, they might get lost. So, simply put, fewer is better. After all, someone has to keep track of those minions. Each minion spawned from the machine can work a set amount of units, and then they are "spent" (there exists a recycling machine, but it is hidden in a deep, dark forest somewhere).

The machine itself creates a given number of minions whose work capacities are normally distributed with both μ and σ unknown. The number of minions the machine can spawn in a given time interval is Poisson distributed with intensity λ, also unknown.

What we are interested in is knowing the minimum amount of times Ruben would have to spawn a minion to be sure that all the work gets done. You are given a list of how many work units each of the N minions can work for. The machine will break down completely after having spawned N minions. The machine lets you choose which minion you want spawn next from the list of possible minions, but you can only spawn each once. All minions are unique in their own ways, but they might still have the same work capacity.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each of the following T cases then consist of two lines. The first line has two integers: W, the number of work units Ruben needs completed, and M, the number of minions the machine can spawn. Then follows a line with M integers C_i, representing how many work units each minion can complete.

Output specifications

Output the minimum number of minions needed to complete the workload W, or output "`no rest for Ruben`" (without the quotes). Please note that you need a capital r in Ruben's name.

Notes and Constraints

- $0 < T \leq 50$
- $0 < W \leq 10000$
- $0 < M \leq 100$
- $0 < C_i \leq 100$

Sample input

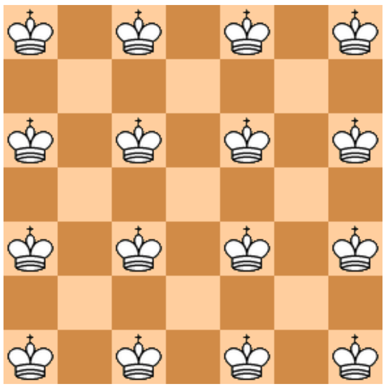
```
4
4 5
1 2 4 100 3
10 10
1 1 1 1 1 1 1 1 1 1
20 10
9 1 1 1 1 1 1 1 1 9
100 5
81 1 2 1 4
```

Output for sample input

```
1
10
4
no rest for Ruben
```

Problem F

Kings on a Chessboard



You are given a chessboard of size $x \times y$ and k identical kings, and are asked to place all the kings on the board such that no two kings can attack each other. Two kings can attack each other if they are horizontally, vertically or diagonally adjacent.

Write a computer program that calculates the number of possible arrangements of the k kings on the given chessboard. Since the number of feasible arrangements may be large, reduce the number modulo 1,000,000,007.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each of the following T lines consists of three integers x, y and k, separated by one space.

Output specifications

For each test case, output the number of possibilities modulo 1,000,000,007.

Notes and Constraints

- $0 < T \leq 50$
- $2 \leq x, y \leq 15$
- $1 \leq k \leq x \cdot y$

Sample input

```
4
8 8 1
7 7 16
7 7 7
3 7 15
```

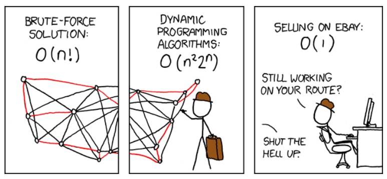
Output for sample input

```

64
1
2484382
0
```

Problem G

Traveling Cellsperson


BRUTE-FORCE SOLUTION: $O(n!)$
DYNAMIC PROGRAMMING ALGORITHMS: $O(n^2 2^n)$
SELLING ON EBAY: $O(1)$
STILL WORKING ON YOUR ROUTE?
SHUT THE HELL UP.

You have solved every problem from Project Euler in your head. Now it is time for a problem you might have heard of, namely The Traveling Salesperson, whose decision version is NP-complete. We consider the Traveling Salesperson problem in a 2D rectangular grid where every cell can be reached from their neighboring cells (up, down, left and right) and you can visit a cell as many times as you like (though, most of the cells aren't that interesting, so you might prefer not to visit them a lot).

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Then follow two integers X and Y, marking the width and height of the grid, respectively. Then follow Y lines with X characters, where the character 'C' is a cell and the character 'S' is the starting point.

Output specifications

For each test case, output the minimum number of steps required to make a full roundtrip of the grid, starting and ending at S, and visiting each cell at least once.

Since you realize that this won't lead anywhere, finish off the output with "LOL" (without quotes) on a line of its own (one per run, not per test case).

Notes and Constraints

- $0 < T \leq 50$
- $0 < X \leq 100$
- $0 < Y \leq 100$
- All characters in a test case are 'C', except for exactly one, which is 'S'.

Sample input

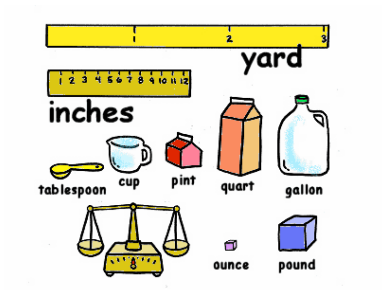
```
1
4 4
CCCC
CCCC
CSCC
CCCC
```

Output for sample input

```
16
LOL
```

Problem H

Dimensions



You and your friend Christian have decided to take a vacation year and travel all around the world to see magnificent places, meet wonderful people, and experience new cultures. Unfortunately, with different cultures come difficult differences. And the hardest differences engineers like you know of are unit differences. Why are people using miles, firkins, microfortnights, candlepowers, boisseaux, foes, pints, kWh and not to mention degrees Celsius when they instead would be much happier using the standard and beloved SI units of metres, kilograms, seconds, amperes, kelvins and candelas?

Quantity	**Name**	**Symbol**
length	metre	m
mass	kilogram	kg
time	second	s
electric current	ampere	A
temperature	kelvin	K
luminous intensity	candela	cd

Table 1: List of SI units

In fact, you love the SI units so much that you refuse to use any other units. Derived units like the joule (J), the newton (N), and the ohm (Ω) are perfectly expressible in their equivalent SI units of kg m^2 / s^2, kg m / s^2, and kg m^2 / s^3 A^2, respectively. So during your travel, you record all units you come across, along with their definitions. Of course, some definitions are depending on other definitions, like Pa = N / m^2.

With your definitions ready at hand, you don't have to put up with such nonsense as 60 firkins / microfortnights or 63 km / h anymore, since you can always convert them SI units. Even calculations like 100 m + 1.3 km and 7 N · 8 Ω become a breeze to you.

Input specifications

The following syntax is given ('?' denotes "zero or one", '+' denotes "one or more", and '*' denotes "zero or more"):

power ::= { integer larger than 1 }
unit ::= { upper or lower case English letter }+
dimension ::= *unit* ['^' *power*]?
size ::= {any floating-point number} [' ' *dimension*]* ['/' [' ' *dimension*]+]?
operator ::= '+' OR '−' OR '*'

expression ::= *size* ' ' *operator* ' ' *size*
unit definition ::= *unit* '=' *size*

All *unit*s have lenghts less than 10, and in a *size*, no *unit*s are repeated.

The first line of the input consists of a single integer U, the number of new units. Then follow U lines with new *unit definition*s. After the unit definitions follows a line with a single integer N, and then N lines with either an *expression* or a *size*.

In the input, all *power*s are less than 5, and all *unit*s are SI unit symbols or previously defined units.

Output specifications

For each computation, output one line with the answer to the expression, or the size itself, converted to SI units. If the answer cannot be computed, output "`Incompatible`" (without the quotes). The answer should be formatted as a *size*, with the following clarifications:

- Units can be written in any order, but keep them on the correct side of the division sign.
- Do not output a unit if its exponent is 0, or its exponent if it is equal to 1.
- Output one space between each unit and between units and the division sign, `\`.
- Do not output any spaces before or after the exponent sign, `^`.

Notes and Constraints

- $0 < U \leq 100$
- $0 < N \leq 1000$
- Units are case sensitive.
- No lines are longer than 140 characters.
- No input, output or part of any computation have size of absolute value above 10^{100}.
- No part of any computation will yield division by 0.
- Any output with a relative or absolute error of 10^{-6} is accepted.

Sample input

```
4
km = 1000 m
h = 3600 s
J = 1 kg m^2 / s^2
X = 3 m^2 kg s / A K cd^4
4
100 m + 1.3 km
63 km / h
1E5 J * 0.003 h^2 / km^2
1 J - 2 X
```

Output for sample input

```
1400.0 m
17.5 m / s
3888.0 kg
Incompatible
```

Problem I

Space Travel

The year is 2014. Mankind has come a long way since that crucial day in 2013 when genius students wrote those groundbreaking programs at IDI Open. Travelling in space has become as normal as feeding your personal crocodile; something you never do more than once...

You want to do something about this, and have identified the problem with space travel to it being too time consuming. You therefore decide to write a program that finds the optimal (shortest in time) travel route between two points in three dimensional space.

This is made somewhat more complex due to the existence of worm tubes. A space traveller can enter a worm tube at any point (on it) and leave at any point. This process takes no time at all. A worm tube is modelled as a line segment in a three dimensional space.

Other than in worm tubes, travel time is proportional to the distance travelled.

Input specifications

The first line contains T, the number of test cases that follow. Each test case starts with a line containing an integer N, the number of worm tubes in space. Then follows a line containing three integers s_x, s_y and s_z, the starting point for the route you're going to find the optimal route for. Then follow a line containing the corresponding end point, e_x, e_y and e_z.

After that follow N lines, describing the N worm tubes. Each worm tube is described by 6 integers s_{xi}, s_{yi}, s_{zi}, e_{xi}, e_{yi} and e_{zi}, describing the start and end point of that tube.

Output specifications

For each test case, output a single floating point number; the distance travelled outside of worm tubes in the optimal route.

Notes and Constraints

- $0 < T \leq 50$
- $0 \leq N \leq 50$
- $0 < p_x, p_y, p_z \leq 1000$, for all s and e.
- The two end points of a worm tube are not equal.
- Any output with a relative or absolute error of 10^{-6} is accepted.

Sample input

```
2
0
10 12 15
9 11 16
2
100 100 100
123 126 129
102 109 103 110 120 113
108 121 104 120 125 122
```

Output for sample input

```
1.7320508075688772
21.56720748874348
```

Problem J

C.S.I.: P15

You have been cast as the computer genius hero-of-the-day for the season finale of the show C.S.I.: P15 (coming this fall). Somewhat unsurprisingly, there is that camera feed that needs to be analyzed. The camera in question is recording pictures in HD-9000 quality with extra regression and the stream is then internally matched by a re-inverted isomorphic bit coefficient matrix, then plasma shifted five times for good measure. You then view the feed through Netscape Navigator 4 Platinum Edition. (Note that "internally" is just fancy talk for "inside the camera".)

Unfortunately, a saboteur turned on ASCII mode on the camera and set the camera in picture burst mode. So now all you have is a bunch of still ASCII images. And now, for reasons that will be revealed later in the show, you are to design and implement a deterministic algorithm for counting the number of flowers and birds in a given still image. The pictures always include the ground, which will show up as a contiguous row of '`=`' characters. The ground will always be the bottom-most row of "ASCII pixels". There will never be anything else on that row (though, on one of the pictures taken before the sabotage there is a stray electron that a someone will accidentally find by zooming in too far, but that is for a later episode).

Air is marked in the feed as a '`.`' (a dot). The ground is the last line of the feed, and it looks like this: '`===========`'. A flower is defined as any 8-connected component (meaning ...) which consists of characters from the set {'`|`', '`/`', '`\`', '`-`', '`@`'}, and which is also connected to the ground. A bird is an occurence of '`/\/\`', surrounded exclusively by air, or by the edges of the image. So if you see something that looks like a bird on the ground, it is a flower (possibly an ex-parrot, but that is also a flower for our purposes).

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each of the following T cases then begins with a line of two integers separated by a space, the height H and width W, and ends with H lines describing the picture. Each line of the picture has exactly W characters. All lines but the last consist of only the following characters: {'`.`', '`|`', '`/`', '`\`', '`-`', '`@`'}. The last line consists of '`=`' characters only.

Output specifications

For each test case, output two lines. If the number of flowers is F and the number of birds is B, the output should read

```
Flowers: F
Birds: B
```

Notes and Constraints

- $0 < T \leq 100$
- $0 < W \leq 30$
- $0 < H \leq 30$

Sample input

```
1
12 28
............................
............................
\@/.../\/\..../\/\..........
.|..........................
.|....\@/.........../\/\....
.|.....|.............|......
.|.....|.............|......
.|.....|..\@/....\@/.|......
.|.....|....\..../...|.|-|..
.|.....|.....\../....|.|.|..
.|.....|......\/.....|.|.|..
============================
```

Output for sample input

```
Flowers: 5
Birds: 2
```