Colombian Collegiate Programming League
CCPL 2014

Contest 10 – September 20

Problems

This set contains 10 problems; pages 1 to 18.

(Borrowed from several sources online.)

Official site `http://programmingleague.org`

Follow us on Twitter `@CCPL2003`

A - It Can Be Arranged

Source file name: `arranged.c`, `arranged.cpp`, *or* `arranged.java`

Every year, several universities arrange inter-university national programming contests. ACM ICPC Dhaka site regional competition is held every year in Dhaka and one or two teams are chosen for ACM ICPC World Finals.

By observing these, MMR (Mission Maker Rahman) has made a plan to open a programming school. In that school, N courses are taught. Each course is taught every day (otherwise, programmers may forget DP while learning computational geometry!). You will be given the starting time A_i and finishing time B_i (inclusive) of each course i ($1 \leq i \leq N$). You will be also given the number of students registered for each course, S_i ($1 \leq i \leq N$). You can safely assume no student has registered to two different courses. MMR wants to hire some rooms of a building, named *Sentinel Tower*, for running that school. Each room of Sentinel Tower has a capacity to hold as much as M students. The programmers (students) are very restless and a little bit filthy! As a result, when $course_i$ is taken in a class room, after the class is finished, it takes $clean_{ij}$ time to clean the room to make it tidy for starting teaching $course_j$ immediately just after course i in the same room.

Your job is to help MMR to decide the minimum number of rooms need to be hired to run the programming school.

Input

Input starts with an integer T ($T \leq 100$) denoting the number of test cases. Each case starts with two integers N ($1 \leq N \leq 100$), number of courses and M ($1 \leq M \leq 10000$), capacity of a room. Next N lines will contain three integers A_i, B_i ($0 \leq A_i \leq B_i \leq 10000000$) and S_i ($1 \leq S_i \leq 10000$), starting and finishing time of a course. Next N lines will contain the clean time matrix, where the ith row will contain N integers $clean_{ij}$ ($1 \leq i \leq N$, $1 \leq j \leq N$, $0 \leq clean_{ij} \leq 10000000$, $clean_{ii} = 0$).

The input must be read from standard input.

Output

For each case, print the test case number, starting from 1, and the answer, minimum number of rooms needed to be hired.

The output must be written to standard output.

Sample Input	**Sample Output**
3 1 5 1 60 12 0 4 1 1 100 10 50 130 3 150 200 15 80 170 7 0 2 3 4 5 0 7 8 9 10 0 12 13 14 15 0 2 1 1 10 1 12 20 1 0 2 5 0	Case 1: 3 Case 2: 22 Case 3: 2

B - Glass Beads

Source file name: `beads.c`, `beads.cpp`, *or* `beads.java`

Once upon a time there was a famous actress. As you may expect, she played mostly Antique Comedies most of all. All the people loved her. But she was not interested in the crowds. Her big hobby were beads of any kind. Many bead makers were working for her and they manufactured new necklaces and bracelets every day. One day she called her main *Inspector of Bead Makers* (IBM) and told him she wanted a very long and special necklace.

The necklace should be made of glass beads of different sizes connected to each other but without any thread running through the beads, so that means the beads can be disconnected at any point. The actress chose the succession of beads she wants to have and the IBM promised to make the necklace. But then he realized a problem. The joint between two neighbouring beads is not very robust so it is possible that the necklace will get torn by its own weight. The situation becomes even worse when the necklace is disjoined. Moreover, the point of disconnection is very important. If there are small beads at the beginning, the possibility of tearing is much higher than if there were large beads. IBM wants to test the robustness of a necklace so he needs a program that will be able to determine the worst possible point of disjoining the beads.

The description of the necklace is a string $A = a_1 a_2 \ldots a_m$ specifying sizes of the particular beads, where the last character a_m is considered to precede character a_1 in circular fashion.

The disjoint point i is said to be worse than the disjoint point j if and only if the string

$$a_i a_{i+1} \ldots a_n a_1 \ldots a_{i-1}$$

is lexicografically smaller than the string $a_j a_{j+1} \ldots a_n a_1 \ldots a_{j-1}$. String $a_1 a_2 \ldots a_n$ is lexicografically smaller than the string $b_1 b_2 \ldots b_n$ if and only if there exists an integer $i, i \leq n$, so that $a_j = b_j$, for each j, $1 \leq j < i$ and $a_i < b_i$.

Input

The input consists of N cases. The first line of the input contains only positive integer N. Then follow the cases. Each case consists of exactly one line containing necklace description. Maximal length of each description is 10000 characters. Each bead is represented by a lowercase character of the english alphabet (a-z), where $a < b < \cdots < z$.

The input must be read from standard input.

Output

For each case, print exactly one line containing only one integer: number of the bead which is the first at the worst possible disjoining, i.e., such i, that the string $A[i]$ is lexicographically smallest among all the n possible disjoinings of a necklace. If there are more than one solution, print the one with the lowest i.

The output must be written to standard output.

Sample Input	**Sample Output**
4 helloworld amandamanda dontcallmebfu aaabaaa	10 11 6 5

C - Count the Trees

Source file name: `count.c`, `count.cpp`, *or* `count.java`

Another common social inability is known as ACM (Abnormally Compulsive Meditation). This psychological disorder is somewhat common among programmers. It can be described as the temporary (although frequent) loss of the faculty of speech when the whole power of the brain is applied to something extremely interesting or challenging.

Juan is a very gifted programmer, and has a severe case of ACM (he even participated in an ACM world championship a few months ago). Lately, his loved ones are worried about him, because he has found a new exciting problem to exercise his intellectual powers, and he has been speechless for several weeks now. The problem is the determination of the number of different labeled binary trees that can be built using exactly n different elements.

For example, given one element A, just one binary tree can be formed (using A as the root of the tree). With two elements, A and B, four different binary trees can be created, as shown in the figure.

If you are able to provide a solution for this problem, Juan will be able to talk again, and his friends and family will be forever grateful.

Input

The input will consist of several input cases, one per line. Each input case will be specified by the number n ($1 \leq n \leq 300$) of different elements that must be used to form the trees. A number 0 will mark the end of input and is not to be processed.

The input must be read from standard input.

Output

For each input case print the number of binary trees that can be built using the n elements, followed by a newline character.

The output must be written to standard output.

Sample Input	**Sample Output**
1 2 10 25 0	1 4 60949324800 7541467185233920829627584924876800000

D - Mixing Colours

Source file name: `mixing.c`, `mixing.cpp`, *or* `mixing.java`

Frank lives in London and he really likes games and maths. Lately, he has been playing a game on his mobile phone. It is simple, a sequence of coloured tokens is provided and each turn the player has to combine a pair of adjacent tokens to obtain a new token of certain colour. This process is repeated until the whole sequence results in only one final token. The problem is that not every pair of colours can be merged. There is a set of rules describing the allowed combinations. For example, given the following rules

```
blue   + yellow -> green
yellow + red    -> orange
blue   + orange -> brown
```

and the sequence (`blue`,`yellow`,`red`) the game could be finished obtaining a `brown` token after two steps:

```
(blue,yellow,red) -> (blue,orange) -> (brown).
```

Frank is now in Valencia attending a famous programming contest and he is waiting for the tram to go to the university. Meanwhile, he is playing this game to fill time but the sun is shining so bright that he can not properly see the screen of his phone. He has some certainty about the possible colour of each token and he is wondering what will be the resulting colour following the most likely play of the game according to his estimation of the input sequence. Given Frank's estimation of two colours `A` and `B`, and the combination `A + B -> C`, the certainty of the obtained colour `C` is computed as $cer(\texttt{C}) = cer(\texttt{A}) \cdot cer(\texttt{B})$.

Input

The first line contains R ($0 < R \leq 100$) the number of rules describing the allowed combinations of colours. Each of the following R lines contains three strings s_1, s_2, and s_3 representing the combination $s_1 + s_2 \rightarrow s_3$. Next line shows the number of test cases T. For each test case an integer C indicates the length of the input sequence of tokens ($0 < C \leq 500$). Each of the next C lines describes a token, it contains a list of pairs $(k, cer(k))$ providing a colour k and its corresponding certainty ($0 < cer(k) \leq 1.0$). The list ends with the word `END`. The sum of the certainties for a certain token always is 1.0. Every colour in a test case has appeared first in the rules describing the game.

The input must be read from standard input.

Output

For each test case, print the resulting colour of the most likely play of the game. If there is no possible combination to finish the game then print `GAMEOVER`.

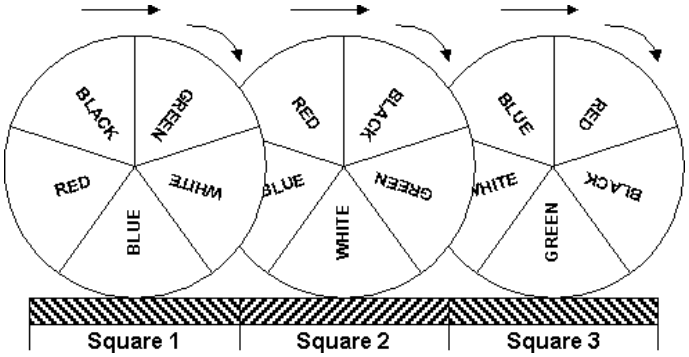
The output must be written to standard output.

Sample Input	**Sample Output**
7 Blue Yellow Green Yellow Red Orange Green Red Yellow White Red Pink Pink Black Red Orange Red Red Yellow Orange Yellow 3 2 Red 0.7 Orange 0.3 END Yellow 1.0 END 4 Blue 0.6 Green 0.4 END Red 0.2 Orange 0.6 Yellow 0.2 END White 0.9 Yellow 0.1 END Red 0.5 Black 0.5 END 2 Blue 1.0 END Orange 1.0 END	Orange Yellow GAMEOVER

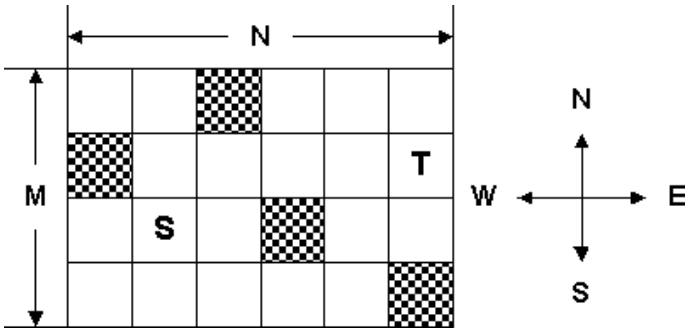
E - The Monocycle

Source file name: `monocycle.c`, `monocycle.cpp`, *or* `monocycle.java`

A monocycle is a cycle that runs on one wheel and the one we will be considering is a bit more special. It has a solid wheel colored with five different colors as shown in the figure:



The colored segments make equal angles (72^o) at the center. A monocyclist rides this cycle on an $M \times N$ grid of square tiles. The tiles have such size that moving forward from the center of one tile to that of the next one makes the wheel rotate exactly 72^o around its own center. The effect is shown in the above figure. When the wheel is at the center of square 1, the midpoint of the periphery of its blue segment is in touch with the ground. But when the wheel moves forward to the center of the next square (square 2) the midpoint of its white segment touches the ground.



Some of the squares of the grid are blocked and hence the cyclist cannot move to them. The cyclist starts from some square and tries to move to a target square in minimum amount of time. From any square either he moves forward to the next square or he remains in the same square but turns 90^o left or right. Each of these actions requires exactly 1 second to execute. He always starts his ride facing north and with the midpoint of the green segment of his wheel touching the ground. In the target square, too, the green segment must be touching the ground but he does not care about the direction he will be facing.

Before he starts his ride, please help him find out whether the destination is reachable and if so the minimum amount of time he will require to reach it.

Input

The input may contain multiple test cases.

The first line of each test case contains two integers M and N ($1 \leq M, N \leq 25$) giving the dimensions of the grid. Then follows the description of the grid in M lines of N characters each. The character '#' will indicate a blocked square, all other squares are free. The starting location of the cyclist is marked by 'S', and the target is marked by 'T'. The input terminates with two zeros for M and N.

The input must be read from standard input.

Output

For each test case in the input first print the test case number on a separate line as shown in the sample output. If the target location can be reached by the cyclist print the minimum amount of time (in seconds) required to reach it exactly in the format shown in the sample output, otherwise, print '`destination not reachable`'.

Print a blank line between two successive test cases.

The output must be written to standard output.

Sample Input	**Sample Output**

`1 3`
`S#T`
`10 10`
`#S.......#`
`#..#.##.##`
`#.##.##.##`
`.#....##.#`
`##.##..#.#`
`#..#.##...`
`#......##.`
`..##.##...`
`#.###...#.`
`#.....###T`
`0 0`
	`Case #1`
	`destination not reachable`
	
	`Case #2`
	`minimum time = 49 sec`

F - Nails

Source file name: `nails.c`, `nails.cpp`, *or* `nails.java`

Arash is tired of hard working, so he wants to surround some nails on the wall of his room by a rubber ribbon to make fun of it! Now, he wants to know what will be the final length of the rubber ribbon after surrounding the nails. You must assume that the radius of nails and rubber ribbon is negligible.

Input

The first line of input gives the number of cases, N. Then N test cases will follow. Each test case starts with a line containing two integers, the initial length of rubber ribbon and the number of nails $0 < n \leq 100$, respectively. Each of next n lines contains two integers denoting the location of a nail. There will be a blank line after each test-case.

The input must be read from standard input.

Output

Your program must output the final length of rubber ribbon rounded up to 5 decimal digits.

The output must be written to standard output.

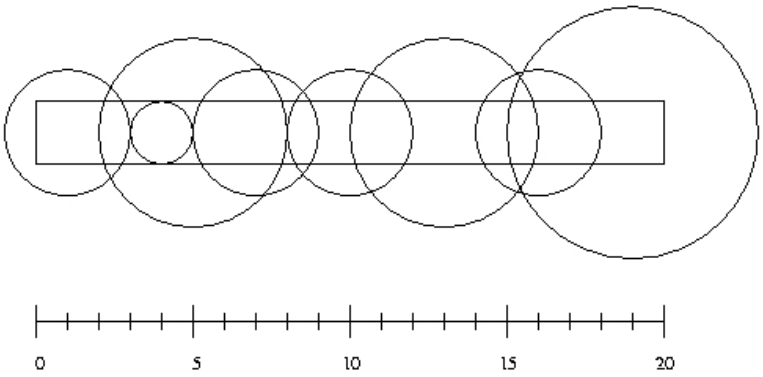
Sample Input	**Sample Output**
2 2 4 0 0 0 1 1 0 1 1 5 4 0 0 0 1 1 0 1 1	4.00000 5.00000

G - Watering Grass

Source file name: `grass.c`, `grass.cpp`, *or* `grass.java`

Consider n sprinklers that are installed in a horizontal strip of grass l meters long and w meters wide. Each sprinkler is installed at the horizontal center line of the strip. For each sprinkler we are given its position as the distance from the left end of the center line and its radius of operation.

What is the minimum number of sprinklers to turn on in order to water the entire strip of grass?



Input

Input consists of a number of cases. The first line for each case contains integer numbers n, l, and w with $n \leq 10000$. The next n lines contain two integers giving the position of a sprinkler and its radius of operation. (The picture above illustrates the first case from the sample input.)

The input must be read from standard input.

Output

For each test case output the minimum number of sprinklers needed to water the entire strip of grass. If it is impossible to water the entire strip output -1.

The output must be written to standard output.

Sample Input	**Sample Output**
8 20 2 5 3 4 1 1 2 7 2 10 2 13 3 16 2 19 4 3 10 1 3 5 9 3 6 1 3 10 1 5 3 1 1 9 1	6 2 -1

H - Love Calculator

Source file name: `love.c`, `love.cpp`, *or* `love.java`

One day I asked Saima that how much she loves me. Her answer was "71.43%". I was surprised as well as shocked by her answer. I could not understand why she didn't say 100%, and why she said a particular and peculiar fraction like 71.43. Looking at my surprised, shocked, and nervous face she burst out laughing and told me that she loves me more than anything in this universe and it was nothing but a silly and funny love calculation. Then she described me the calculation. In this problem you will have to write a program so that anyone can calculate love between any two persons very quickly (of course a very silly game).

You will be given two names. These two names can have white space or some other non-alphabetical characters like $, @, &, %, etc. But only the alphabets from a to z or A to Z will participate in love calculation. Each alphabet has a particular value. The values are from 1 to 26 in ascending order of the alphabets. Its like this, $a = 1$, $b = 2$, $c = 3$, ..., $z = 26$. Both upper case and lower case holds the same values. Then make the sum of these numbers until it comes in one digit. For example, consider a name bcz. Here, $b = 2$, $c = 3$, and $z = 26$. So, the sum is $(2 + 3 + 26) = 31 = (3 + 1) = 4$. Then the ratio of these two numbers in percentage will be the result. Remember that the result cannot be more than 100%; take the ratio carefully to avoid this problem.

Input

Your input will be two names, each one on a single line. Each name holds not more than 25 characters. End of input indicates the end of input.

The input must be read from standard input.

Output

For each pair of names your program will have to print a line indicating the love between those two persons. The result is a floating point value rounded up to two decimal places.

The output must be written to standard output.

Sample Input	Sample Output
saima shanto Pakistan %%% @$% $	71.43 % 0.00 % 100.00 %

I - Interval Product

Source file name: `interval.c`, `interval.cpp`, *or* `interval.java`

It's normal to feel worried and tense the day before a programming contest. To relax, you went out for a drink with some friends in a nearby pub. To keep your mind sharp for the next day, you decided to play the following game. To start, your friends will give you a sequence of N integers $X_1, X_2, \cdots, X_N$. Then, there will be K rounds; at each round, your friends will issue a command, which can be:

- a *change command*, when your friends want to change one of the values in the sequence; or
- a *product command*, when your friends give you two values I, J and ask you if the product $X_I \times X_{I+1} \times \cdots \times X_J$ is positive, negative, or zero.

Since you are at a pub, it was decided that the penalty for a wrong answer is to drink a pint of beer. You are worried this could affect you negatively at the next day's contest and you don't want to check if Ballmer's peak theory is correct. Fortunately, your friends gave you the right to use your notebook. Since you trust more your coding skills than your math, you decided to write a program to help you in the game.

Input

Each test case is described using several lines. The first line contains two integers N and K, indicating respectively the number of elements in the sequence and the number of rounds of the game ($1 \leq N, K \leq 10^5$). The second line contains N integers X_i that represent the initial values of the sequence ($-100 \leq X_i \leq 100$, for $i = 1, \ldots, N$). Each of the next K lines describes a command and starts with an uppercase letter that is either 'C' or 'P'. If the letter is 'C', the line describes a change command, and the letter is followed by two integers I and V indicating that X_I must receive the value V ($1 \leq I \leq N$) and ($-100 \leq V \leq 100$). If the letter is 'P', the line describes a product command, and the letter is followed by two integers I and J indicating that the product from X_I to X_J, inclusive must be calculated ($1 \leq I \leq J \leq N$). Within each test case there is at least one product command.

The input must be read from standard input.

Output

For each test case output a line with a string representing the result of all the product commands in the test case. The i-th character of the string represents the result of the i-th product command. If the result of the command is positive the character must be '+' (plus), if the result is negative the character must be '−' (minus), and if the result is zero the character must be '0' (zero).

The output must be written to standard output.

Sample Input	**Sample Output**
4 6 -2 6 0 -1 C 1 10 P 1 4 C 3 7 P 2 2 C 4 -5 P 1 4 5 9 1 5 -2 4 3 P 1 2 P 1 5 C 4 -5 P 1 5 P 4 5 C 3 0 P 1 5 C 4 -5 C 4 -5	0+- +-+-0

J - Binary Tree

Source file name: `tree.c`, `tree.cpp`, *or* `tree.java`

Binary Tree is a tree data structure where each node has at most two children, usually they are distinguished as *left* and *right* child. And the node having the children are called parent of those children. An instruction string is a string consisting of the letters L, R, and U. L stands for *Left*, R for *Right*, and U for *Up*. Meaning of these will be clear shortly.

One day I have drawn an infinitely large Binary Tree. In this tree each node has exactly two children (*left child* and *right child*) and each of them has a parent. For this problem, we will consider the *parent of the root* is the root itself. I put a pen in the root and follow the instruction string S. That is, we look at the first character if it says L we go to left child, if it says R we go to right child, and if it says U then to the parent. If we receive a U instruction at root we just end up at root since we assumed parent of the root is root itself.

Now we have another instruction string T. Starting from the node where we are after following the instruction string S, we will follow the instruction string T. But this time, if we wish we may skip any instruction in the string T (possibly discarding all of them). You have to tell me how many different nodes I can end up after following instruction string T (skipping as many instructions as I wish).

For example, suppose: $S = L$ and $T = LU$. Our answer is 3. Following S we will end up at the left child of the root. Now, when we follow T, there may be 4 cases:

(i) Skipping all letters: we will be at the same node where we are.

(ii) Skipping L and following U: we will be at the root.

(iii) Following L and Skipping U: we will be at the left child of current node.

(iv) Following both L and U: we will be at the same node as in case (i).

Since there are 3 different nodes we can end up at after following T, the answer is 3.

Input

First line of the test file contains an integer N ($N \leq 15$) denoting number of test cases. Hence follow N test cases. Each test case consists of two non empty strings. First line will contain instruction string S and the second line will contain the instruction string T. You may assume that there will not be any letter other than L, R, or U in these strings. Length of the strings will not be greater than 100000.

The input must be read from standard input.

Output

For each test case print the case number followed by the number of nodes we can end up finally. Since the answer may be large, you have to give the answer modulo 21092013.

The output must be written to standard output.

Sample Input	**Sample Output**
2 L LU L L	Case 1: 3 Case 2: 2