A Problem with Strings

Input file: stdin
Output file: stdout

Today, we will be unraveling the mysteries of one of the most twisted and curly beasts in all existence knots! The problem here is simple: given the picture of a string laid out curled on a table, you are to determine if it will curl up into a knot when the ends are pulled out.

```
..............
...+------+....
...|......|....
...|...+--H----
...|...|..|....
---I---H--+....
...|...|.......
...+---+.......
..............
```

The input will consist of a diagram of one single continuous string described in a two-dimensional grid of characters such as shown above. The '−' and '|' characters represent a horizontal and vertical segment of the string, respectively. '+' characters represent corners where the string turns at right angles on the table. 'I' or 'H' characters represent locations where parts of the strings cross:

- 'H' represents locations where the horizontal string passes over the vertical string
- 'I' represents locations where the horizontal string passes under the vertical string

The dot character, '.', obviously, represents empty spaces of the table unoccupied by the string. Two horizontally adjacent non-empty spaces of the table are connected by the string iff neither of them are '|' characters. Similarly, vertically adjacent non-empty spaces are connected by the string iff neither of them are '-' characters. Inputs will be such that every '+' character will represent a proper corner where the string turns at a unambiguously at a right angle: formally, every '+' character will be connected to exactly one vertically adjacent and exactly one horizontally adjacent space. Moreover, to further simplify matters, you can assume that the only characters along the border of the given diagram, other than dots, will represent endpoints of the string. In short, the input will unambiguously specify a valid piece of string starting and ending at the border of the input diagram. Finally, assume that the string has negligible width and is made of a very smooth material, so that parts of the string can easily slide over each other with negligible friction.

Input

The input will consist of at most 80 test cases. Each test case will start with a single line containing two integers, r and c, indicating that the size of the diagram for that case is of r rows and c columns. This line will be followed by r more lines, each with exactly c characters, with characters in each line representing a row of the diagram. You may assume that $2 \leq r, c \leq 120$.

Output

The output should consist of exactly one line for each test case in the format Case c: Response, where c is the test case serial number, starting from 1, and Response is either the string straightened or knotted (without the quotes) depending on whether the string will straighten out or coil up into a knot, respectively. See the sample for further clarifications.

Sample input and output

stdin	stdout
9 15 ....+------+.... ...|......|.... ...|...+--H---- ...|...|..|.... ---I---H--+.... ...|...|....... ...+---+....... .... 9 15 ....+------+.... ...|......|.... ...|...+--I---- ...|...|..|.... ---I---H--+.... ...|...|....... ...+---+.......	Case 1: knotted Case 2: straightened

B Bonus

Input file: stdin
Output file: stdout

Kinku is a project manager in his company. Recently, the executives decided to give bonus to everyone in his team, so Kinku was asked to give a list to them how much bonus each should get.

To do this, he selected some pairs of members, and analyzed who should get more bonus, and how more much he should get (at least). That is, if employee Tom is better than Jerry, and Tom should get at least 300$ more than Jerry, then, if Jerry gets 650$ bonus, Tom has to get atleast 950$ bonus. It should also be noted that, every one will get at least a minimum, no one will go empty handed.

Given all the comparisons, you have to find the minimum total bonus to be divided among the employees.

Input

First line contains T, the number of test cases.

Each test case starts with three integers $N(1 \leq N \leq 100)$, $M(0 \leq M \leq \frac{N \times (N-1)}{2})$ and $L(1 \leq L \leq 10000)$, the number of employees, number of comparisons and minimum bonus. Next M lines each contains three integers, i, $j(1 \leq i, j \leq N$ and $c(0 \leq c \leq 1000)$, referring that $employee_i$ should get at least c\$ more than $employee_j$.

Output

For each test case, output the minimum total bonus on the first line, and the bonus given to each employee in the next line.

If comparisons are inconsistent, and it's not possible distribute bonus among them, output "`Inconsistent analysis.`"(without quotes).

Sample input and output

stdin	stdout
2 4 4 100 1 2 10 2 3 20 1 3 40 2 4 5 3 3 100 1 2 10 2 3 10 3 1 10	460 140 120 100 100 Inconsistent analysis.

C Problem Name

Input file: `stdin`
Output file: `stdout`

Traffic Jam is becoming a major problem in the city of 'Faka' these days. Every day, you need to go to work from home through a one-way road that can be modeled as a series of straight line segments (aka piece-wise linear) of positive length. To be more explicit, suppose there are one way straight line paths named P, Q and R in your way. P starts at your home & ends at the starting point of Q. Q starts from the ending point of P & ends at starting point of R. Similarly, R starts at ending point of Q & ends at your destination. Two non-adjacent line segments in a road will never intersect with each other.

Now, you start from home for work and immediately get irritated by the heavy and annoying traffic jam. So you wish you could fly to your destination in order to avoid spending the whole day on road. Your new vehicle Aerocar comes as the solution to you. With this car, you can leave the road & start to fly from any point (The aerocar has a vertical take-off & landing similar to that of a helicopter) and land back to the road any time. Now, though you would have liked to fly all the way and thus avoiding the traffic, you need to resist the temptation due to the fact that flying requires more fuel than driving. You need 1 unit of fuel to travel every unit distance along the path while F units per unit distance while flying. Now you need to write a program that, given the description of the road, can calculate the minimum possible fuel amount needed to complete your journey.

Input

Each test case starts with a couple of integers $N(1 \leq N \leq 25)$ & $F(2 \leq F \leq 5)$. N is the number of line segments in the road. F is the amount of fuel required for every unit distance traveled while flying. Next $(N+1)$ lines each has two integers, $x(-1000 \leq x \leq 1000)$ & $y(-1000 \leq y \leq 1000)$. Assuming the road is placed on a planar 2d grid, the integers in the i'th line denotes the x and y co-ordinates of the starting point of the i'th line segment while the the end point co-ordinates by the i'th line segment are given at the $(i+1)'$th line. The last test case will be followed by a case with $N = F = 0$ indicating the end of input. This case should not be processed.

Output

For each test case except the last one, print one line of the form "Case $X : Y$", where X is the serial number of output & Y is the minimal possible units of fuel required. Print 3 digits after decimal point for Y. The output will be tested by a special judge program and outputs with precision error smaller than 10^{-3} shall be considered as correct.

Sample input and output

stdin	stdout
2 5 0 0 5 0 10 0 2 2 0 0 2 2 2 -2 3 5 0 0 1 1 -7 0 1 10 0 0	Case 1: 10.000 Case 2: 5.464 Case 3: 22.283

D A New Number System

Input file: stdin
Output file: stdout

As we know, in an n-based number system, there are n different types of digits. In this way, a 1-based number system has only 1 type of digit, the '0'. Here are the rules to interpret 1-based numbers. Each number consists of some space separated blocks of 0. A block may have 1, 2 or more 0s. There is a 'flag' variable associated with each number

- A block with a single 0 sets 'flag' variable to 1
- A block with two 0s sets the 'flag' to 0
- If there are n ($n > 2$) 0s in a block, $n2$ binary digits with the current value of flag is appended to your number.

Note that, the first block of every number will have at most 2 0s. For example, the 1-base number 0 0000 00 000 0 0000 is equivalent to binary 11011.

- 1st block sets the flag to 1.
- 2nd block has 4 0s. So append flag(= 1) $42 = 2$ times (11).
- 3rd block has 2 0s. Set the flag to 0
- 4th block has 3 0s. Append flag(= 0) $3 - 2 = 1$ time (110).
- 5th block has a single 0. Set flag = 1
- 6th and block has 4 0s. Append flag(= 0) $4 - 2 = 2$ times (11011).

The final binary number wont have more than 30 digits. Once, youve completed the process, convert the binary value to decimal & print, youre done!

Input

Input will have at most 100 test cases. Each case consists of a 1-based number as described above. A number may be spanned to multiple lines but a single block will always be in a single line. Termination of a case will be indicated by a single '#' char which will be space-separated from the last digit of your input number. The last case in the input is followed by a '∼' character indicating, end of input.

Output

For each test case, output a single line with the decimal equivalent value of your given 1-based number.

Sample input and output

stdin	stdout
0 0000 00 000 0 0000 # 0 000 # ∼	27 1

E Algebra Express

Input file: stdin
Output file: stdout

Mim is a high school algebra teacher. Just last week, she taught her students how to evaluate an expression. For example, if

$$r = a \times (b + c) - d \times e$$

Given the values of a, b, c, d and e, they can evaluate r.

Now, to test how well her students are able to do these arithmetic, she has to prepare some test. But, she do this every year, and each time she has to make same type of questions. That made her a little bored. She thinks that evaluating an expression only depends on the position of operators and variables, not the variables itself. For example in the expression above, for all possible values of a, b, c, d and e, they will all be equally hard to evaluate.

She has made some generic expressions, and a set of values, which she may use in any order (e.g. one time she may use $a = 2$, $b = 5$, another time she may use $a = 5$, $b = 2$). Now, she wants to know, the number of possible ways to assign these values, so that, the value of the expression lies within a specified limit.

Input

First line contains T, the number of test cases.

Each test case starts with three integers $N(2 \leq N \leq 12)$, l, h $(-10^{12} \leq l \leq h \leq 10^{12})$, the number of variables, lower limit and the upper limit of the acceptable expression values.

Next line contains N integers $a_i(0 \leq a_i \leq 9)$, the value of the variables she will be using in the exam.

Next line contains the expression, where each variable is represented as '`x`'. The expression will be well formed, and will contain only the symbols '+', '-', '*','(',')' and '`x`'.

Output

For each test case, output the number of different assignments, that produce results within the limit $[l, h]$.

Note that, you will be replacing each '`x`' with a value $a_j(1 \leq j \leq N)$. Each a_j have to be used exactly once. Two assignments are considered the same, if, after replacing all '`x`', the expressions look the same.

Sample input and output

stdin	stdout
3 3 1 10 1 2 5 x+x*x 4 1 10 1 2 2 5 (x+x)*(x-x) 4 1 100 1 1 1 1 x*(x-(x*x))	4 4 0