A The Super Powers

We all know the Super Powers of this world and how they manage to get advantages in political warfare or even in other sectors. But this is not a political platform and so we will talk about a different kind of super powers – "The Super Power Numbers". A positive number is said to be super power when it is the power of at least two different positive integers. For example 64 is a super power as $64 = 8^2$ and $64 = 4^3$. You have to write a program that lists all super powers within 1 and 2^{64} -1 (inclusive).

Input

This program has no input.

Output

Print all the Super Power Numbers within 1 and 2^64 -1. Each line contains a single super power number and the numbers are printed in ascending order.

Sample Input	Partial Judge Output
No input for this problem	**1** **16** **64** **81** **256** **512** **.** **.** **.**

Problem Setter: Shahriar Manzoor, Special Thanks: Sohel Hafiz

B Creating Palindrome

You are helping the problem setters of International Creating Palindrome Contest(ICPC) in problem setting for the upcoming contest. In ICPC contestants are given some problems. Each problem is an integer sequence. The goal of the contestants is to convert the sequence into a palindromic integer sequence. A palindromic integer sequence is an integer sequence that is the same sequence when written from forwards or backwards, i.e., of the form {a1, a2, a3, …. , a3, a2, a1}. Some examples of palindromic sequences are {78, 91, 78}, {100}, {10, 20, 20, 10}, {5, 5} etc. But {1, 2, 3, 1}, {10, 20} are not palindromic sequence. You can convert {1, 2, 3, 1} to a palindromic sequence by inserting a 2 at 4^{th} position which will become {1, 2, 3, 2, 1} and you can convert {10, 20} to palindromic sequence by inserting 20 at first or inserting 10 at last.

As contestant will be given problems(Integer sequences) with different difficulties, problem setters are asked to create problems of different difficulties. The difficulty of a problem is represented by an integer, for example, 0, 1, 2, 3,... etc. The difficulty of a problem is D, if at least D insertions of integer required to convert the given sequence into palindromic sequence.

Problem setters are also asked to give problems with difficulty not greater than K. Your job is to determine the difficulty of a given sequence to help the problem setters. If the given sequence is already a palindrome, it cannot be used in the contest. So you have to answer "Too easy" without quotes. If the difficulty of the given sequence is more than K, that problem also cannot be used. In that case you have to answer "Too difficult" without quotes. Otherwise you just have to answer the difficulty.

Input

Input will start with an integer T(1<=T<=100), number of test cases. Each test case starts with two integers N(1<=N<=10000) and K(0<=K<=20), the length of integer sequence and maximum allowed difficulty.

Output

For each test case, output a single line of the form "Case X: D". Here X is the case number. If the given sequence is already a palindromic sequence D will be "Too easy" without quotes, if the difficulty of the given sequence is greater than K D will be "Too difficult", other wise D will be the difficulty of the given sequence. See the sample input and output for exact format.

Sample Input	Sample Output
4 4 2 10 20 30 10 4 2 10 20 30 40 4 3	Case 1: 1 Case 2: Too difficult Case 3: 3 Case 4: Too easy

10 20 30 40 **3 0** **10 20 10**	

Problem setter: Md. Arifuzzaman Arif, Special Thanks: Abdullah Al Mahmud

C Code Feat

Hooray! Agent Bauer has shot the terrorists, blown up the bad guy base, saved the hostages, exposed the moles in the government, prevented an environmental catastrophe, and found homes for three orphaned kittens, all in the span of 19 consecutive hours. But now, he only has 5 hours remaining to deal with his final challenge: an activated nuclear bomb protected by a security code. Can you help him figure out the code and deactivate it? Events occur in real time.

The government hackers at CTU (Counter-Terrorist Unit) have learned some things about the code, but they still haven't quite solved it. They know it's a single, strictly positive, integer. They also know several clues of the form "when divided by X, the remainder is one of {Y1, Y2, Y3, ..., Yk}". There are multiple solutions to these clues, but the code is likely to be one of the smallest ones. So they'd like you to print out the first few solutions, in increasing order.

The world is counting on you!

Input

Input consists of several test cases. Each test case starts with a line containing C, the number of clues (1 <= C <= 9), and S, the number of desired solutions (1 <= S <= 10). The next C lines each start with two integers X (2 <= X) and k (1 <= k <= 100), followed by the k distinct integers Y1, Y2, ..., Yk (0 <= Y1, Y2, ..., Yk < X).

You may assume that the Xs in each test case are pairwise relatively prime (ie, they have no common factor except 1). Also, the product of the Xs will fit into a 32-bit integer.

The last test case is followed by a line containing two zeros.

Output

For each test case, output S lines containing the S smallest positive solutions to the clues, in increasing order.

Print a blank line after the output for each test case.

Sample Input	Sample Output
3 2 2 1 1 5 2 0 3 3 2 1 2 0 0	5 13

Problem Setter: Derek Kisman, Special Thanks: Samee Zahur

D Table Tennis

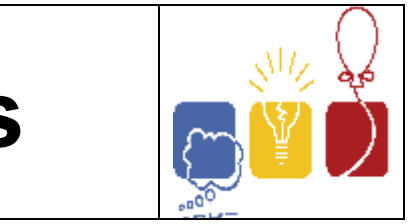
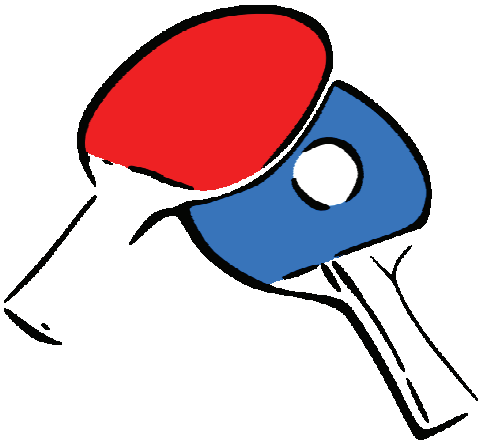


Table tennis is a two/four player sport that originated in England. The scoring system of this game has changed with time. For this problem we will consider the two-player version of the game that abides by the following scoring rules (note that this rule is significantly different from the usual ones) –



- Player 1 makes the first move. The players alternate serve every 5 points. That means serves [1,5] are done by player 1, serves [6, 10] are done by player 2, serves [11, 15] are done by player 1 and so on.
- In each serve, one of the two players wins a point. The first player to reach 21 points wins the game.
- If the scores are 20-20 (deuce), the scores are reset to 15-15.

Given a partial game, the probability of player 1 winning a point on his serve and the probability of player 1 winning a point on the opponent's serve, can you find out the probability of player 1 winning the game?

Input

The first line of input is an integer T(T<1000) that indicates the number of test cases. Each case consists of two lines. The first line is a string consisting of the letters "W" and "L" only. The length of this string is non-negative and can have a maximum value of 100. The string gives the status of the game so far. If the **i^{th}** character (1 based) is "W", that means player 1 won the **i^{th}** point. Similarly, "L" indicates that the player 1 lost that point. The next line consists of two real numbers, **P1 P2**, in the range [0, 1], with at most 3 digits after the decimal point. P1 is the probability of player 1 winning a point on his serve and P2 is the probability of player 1 winning a point on player 2's serve.

Output

For each case, output the case number first. Then output the probability of player 1 winning the game rounded to 6 decimal places. If the given partial game is impossible according to any of the rules or data, output "-1.000000" instead. Outputs will be checked with special judge, so small precision errors will be ignored. Look at the samples for exact format.

Sample Input	Sample Output
4 WWWWWWWWWWWWWWWWWWWWWW 1.0 0.234 WWWWWWWWWWWWWWWWWWWWWW 0.3 0.99	Case 1: 1.000000 Case 2: 1.000000 Case 3: -1.000000 Case 4: 0.444026

WWWWWWWWWWWWWWWWWWWWWW **1.0 1.0** **WL** **0.7 0.3**	

Problem Setter: Sohel Hafiz, Special Thanks: Samee Zahur

Illustration

Case 1: Player 1 has already won 20 points and next it's his serve. To win the next point (which is also the game point) he has a 1.0 probability of winning

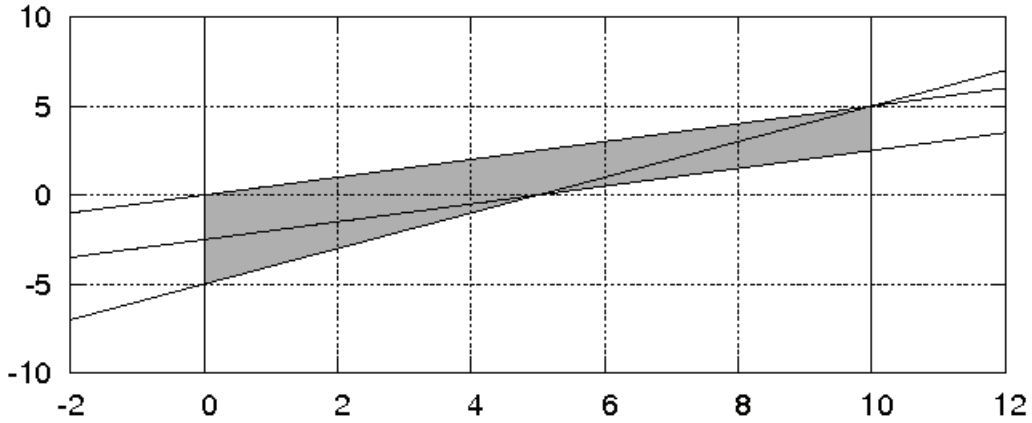
Case 2: Player 1 has already won 21 points and he won the game and so it's 1.000000.

Case 3: The game should stop after 21 points, but the string has an extra "W" and makes this game invalid, so output is -1.000000.

E Bounded by Lines

You are asked to write a program to plot a set of lines. Since, the dimension of the screen is finite, you want to show a portion of the plot.

Lets say, you have some lines, and want to show the portion between x=0 and x=10. Lines are infinitely long, but you only need to consider the portion you are going to show. Find the area bounded by the lines within this region. An area bounded by line may be defined as the set of points, for which, there is at least one line above it, and at least one



line below it.

The shaded area in the graph is the area bounded by lines within the region x=0 and x=10.

Input

Input starts with an integer T(1<= T<=10), the number of test cases. This is followed by T test cases, each starts with an integer N(2<=N<=100000), the number of lines to follow. Next N lines each contain 4 real numbers, x_1, y_1, x_2, y_2,(-10001 < x_1,y_1,x_2,y_2 < 10001) denoting a line between the points (x_1,y_1) and (x_2,y_2).

The description of lines is followed by two integers x_l and x_u (-10001 < x_l, x_u < 10001).

There will be a blank line before each test case.

Output

For each test case, output the area bounded by the lines, within the region $[x_l, x_U]$. You can assume that, the output will never exceed 10^9. Output having difference by no more than 0.001 with the official output shall be considered as correct.

Sample Input	Sample Output
4 2 0 0 5 5 0 0 10 5 0 5 3 0 0 10 5 5 0 10 5 5 0 15 5 0 10 3 0 0 10 5 5 0 10 5 5 0 15 5 0 5 2 0 0 1 1 0 0 -1 1 -1 1	6.250000 31.250000 18.750000 2.000000

Problem Setter: Manzurur Rahman Khan, Special Thanks: Samee Zahur

The image corresponds to the second test case

F Winger Trial

After the great winger Donaldo left his soccer team, coach sir Thelex has found himself in a great fix. The strength of his team is reduced greatly and he needs to find a suitable replacement immediately. The coach selects a number of young wingers from around the world and sets up a trial for them.

The trial will take place on a rectangular shaped field of length **L** meters & width **W** meters. There are **N** robot defenders placed on the field. The defenders do not change their positions but if a winger's distance from a defender is not more than **d** meters, it will automatically tackle him. A robot defender may tackle at most once. On the beginning of the trial, a winger stands on the left edge of the field (across the length) with a soccer ball. Now, his task is to avoid the obstructions of the robot defenders and reach the rightmost edge of the field with the ball. Please tell him the minimum number of tackles he must face in order to reach the opposite end. A player must not go outside the field or he will be disqualified.

Input

Every test case begins with 4 integers, **L (1<=L<=10000)**, **W (1<=W<=10000)**, **N (1<=N<=100)** & **d (1<=d<=1000)**, as described above. Each of the following **N** lines contain 2 integers each, defining the **x** & **y** co-ordinates of a defender. You can consider the co-ordinate of the lower-left corner of the field to be **(0,0)** and upper-right corner to be **(L,W)**. Obviously, all the defenders are located inside the field.

The end of input will be marked by a case with **L=W=N=d=0**. This case should not be processed.

Output

For each test case, print a line in the format, "Case X: Y", where X is the case number & Y is the minimum number of tackles that needs to be dealt with.

Sample Input	Sample Output
500 300 5 100 250 0 250 150 250 300 100 150 400 150 0 0 0 0	Case 1: 1

Problem Setter: Mohammad Mahmudur Rahman, Special Thanks: Manzurur Rahman Khan

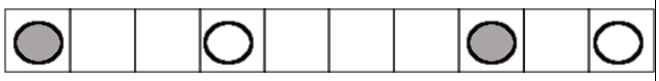
G Left Right

Since you have been participating in programming contests and problem solving for quite a long time, I think most of you know about Alice and Bob :). They are responsible for inventing new games on various occasions. This time, they are back with a new game again.

This new game is played in a one-dimensional board. The board contains ***N*** cells. The cells are numbered from 0 to ***N***-1, where the left most cell is marked as cell 0. Each cell can contain at most one piece. There are two kinds of pieces, gray and white. Alice moves all gray pieces, and bob moves all white ones. The pieces alternate, that is, leftmost piece is gray, next is white, next to that is gray, then it's white again, and so on. There will always be equal number of black and gray pieces.

At each move, the player selects at least one, and at most ***d*** of it's pieces and move each one, either to it's left or to it's right, any number of cells (at least 1). Please note that, in each move, you can move some pieces left, and some pieces right, and also, you can keep some pieces unmoved, as long as, you move at least one of your pieces. But, it can neither jump over other pieces, nor it can move outside the board. The players alternate their moves.

For example, if Alice decides to move the left most gray piece, these two moves are available to her.

	
Moving the gray piece to the left	*Moving the gray piece to the right*

Alice moves first. The game ends, when someone is unable to make any move, and loses the game. You can assume that, both of them play optimally (that is, if it is possible to apply a strategy that will ensure someones win, they will always use that strategy).

Although, in most occasions, they want to know, given the state of the game, who will win that game. But you are not required to find that here. Instead, Alice and Bob wrote a computer program to assist them in playing this game. The program generates the positions of all pieces randomly (the program always generate valid positions, that is no two piece are on the same cell, and the colors of the pieces alternate).

You are asked that, given the size of the board, and the number of gray and white pieces, how many different configurations ensure that Alice will win?

Input

First line of input contains an integer **T(<=100)** the number of test cases. Each of the next T lines contain three integers, **N(1<=N<=10000)**, **K(1<=K<=100**, **K<=N)** and **d(1<=d<=K)**, where **N** is the size of the board, and **K** is the number of pieces, and **d** is the maximum number of piece to move. **K** will always be an even number.

Output

For each test case, output the number configurations, where Alice wins. The result can be very large. So, output the number modulo 1000000007.

Sample Input	Sample Output
5 10 2 1 10 4 2 50 6 3 3642 68 10 5698 32 1	36 182 15874485 367653436 998833190

Problem Setter: Manzurur Rahman Khan, Special Thanks: Samee Zahur

H IBM Fencing

The Indigenous Barrier Makers (IBM) are out in the jungle of the Amazon to make fences for the people living there. Many new inhabitants are coming to stay there so IBM has to build many fences. But all the fences are not of the same type; there are some wooden fences and some steel fences. And the cost of making the steel fence is much higher than the wooden fence. So IBM wants to build wooden fence whenever it is possible to minimize the total cost. But ACM (Association of Consumer-right Monitoring) is more than concerned about the safety of the people and about saving trees.

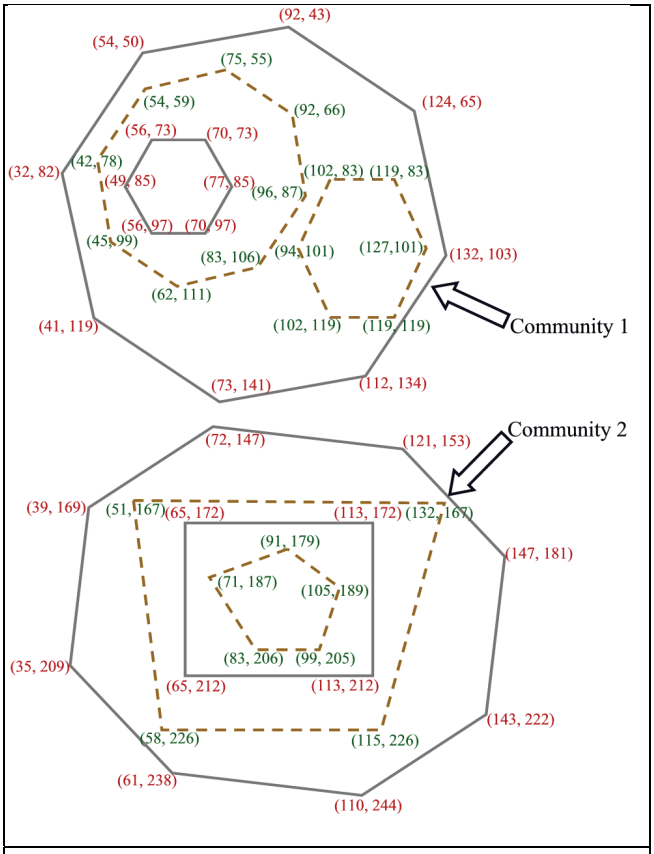


Figure 1: A fence plan. The steel fences are drawn with gray lines and the wooden fences are drawn with brown dashed lines. This figure corresponds to the first sample input and output.

With the adverse effect of climate change around the world, wild fire is a common threat now, so wooden fence is sometimes very risky in places like Amazon. Also make wooden fence we need a lot of woods which eventually destroys a lot of trees. Therefore fences are built in layers for the inhabitants for better protection, safety and privacy. To stay safe as well as to save cost, ACM and IBM decides to build the outermost fence with steel and then a wooden fence and steel fence alternately inwards. Figure 1 shows the top view of such a fence layout. The grey lines indicate that steel fence will be built by IBM along those lines. The brown dashed lines indicate that wooden fences will be built along those lines. Given a fence layout, you will have to help ACM and IBM to find out the total cost for building fences by writing a program. Your solution must be quite efficient.

Input

The input file contains at most 12 sets of inputs. The description of each set is given below:

The first line of each set is an integer N (0<N<501) which denotes how many fences are to be built. Each of the next N lines describes a fence ring. Each fence ring is described as a complete convex polygon. The description starts with an integer M (0<M<1001) which denotes the total number of wall segments that make up the fence ring. This M is followed by M pairs of floating-point numbers (x_1, y_1), (x_2, y_2), (x_3, y_3), …, (x_M, y_M). These integers

denote that the fence is made by connecting the points (x_1, y_1) with (x_2, y_2), (x_2, y_2) with (x_3, y_3), … , (x_{M-1}, y_{M-1}) with (x_M, y_M) and (x_M, y_M) with (x_1, y_1). You can assume that the fence layout will be a valid one – (a) two fence rings will not intersect (b) fence rings will always be described as a convex polygon. You cam also assume that absolute value of all the coordinates will be less than 400000.
Input is terminated by a line containing a single zero.

Output

For each line of input produce 5 lines of output.
The first line contains the serial of output.

The next line reports the total number of communities. A community is comprised by people who share a common outermost boundary.

The third line contains the string "`Total Cost:`"

The fourth and fifth line reports the total cost to build the steel fence and wooden fence respectively. The cost has to be reported million Yuan rounded to the eight digits after the decimal. Assume that the cost of building steel fence is 100 Yuan/unit length and that for Wooden fence is 50 Yuan/unit length.

Print a blank line after output for each set of input. Look at the output for sample input for details. Output difference due to small precision error will be ignored.

Sample Input

```
8

8 112 134 73 141 41 119 32 82 54 50 92 43 124 65 132 103
6 119 119 102 119 94 101 102 83 119 83 127 101
8 83 106 62 111 45 99 42 78 54 59 75 55 92 66 96 87
6 70 97 56 97 49 85 56 73 70 73 77 85
8 143 222 110 244 61 238 35 209 39 169 72 147 121 153 147 181
4 115 226 58 226 51 167 132 167
4 113 212 65 212 65 172 113 172
5 99 205 83 206 71 187 91 179 105 189
2
4 0 0 100 0 100 100 0 100
4 1000 1000 1100 1000 1100 1100 1000 1100
0
```

Output for Sample Input

```
Case 1:
Total Number of Communities 2
Total Cost:
Steel Fence: 0.09047417 Million Yuan
Wooden Fence: 0.03190241 Million Yuan

Case 2:
Total Number of Communities 2
```

```
Total Cost:
Steel Fence: 0.08000000 Million Yuan
Wooden Fence: 0.00000000 Million Yuan
```

Problem Setter: Shahriar Manzoor, Special Thanks: Derek Kisman

I Brother Arif, please feed us!

Brother Arif is a great problemsetter. He loves grid. Also, he loves light. That's why the other problemsetters believe that asking him to throw a feast is always right. Unfortunately Brother Arif himself doesn't think so. Instead of all the naggings and reasons going on around him, he stays in his chair with dreamy eyes and lovingly thinks about newer ways of placing lights inside a 2D grid. Now, it's time to make your stand for the deprived problemsetters' right and force Brother Arif to their pleas.

Now, the problemsetters have all gathered inside a room. The floor of this room is shaped like a 2d Rectangular grid consisting of many cells. Any of the problemsetters can stand on a cell. A cell has enough place for only one people and everyone must occupy exactly one cell at a time. Brother Arif, standing on one of these cells, can move to one of the four adjacent cells just once or he may not move at all. He may not leave the grid, however. The other problemsetters are standing on other cells. With the help of their new telecatching device, they can capture anyone standing at the same row or column irrespective to his distance. So, now you are given the positions of the problemsetters and are asked to figure out if Brother Arif can escape from getting captured (and thus throwing a huge party.)

Input

The input file has multiple test cases. Each test case begins with 3 integers, **R (1<=R<=10000), C (1<=C<=10000) & N (1<=N<=2000)**. **R** &**C** are the number of rows & columns in the grid while **N** denotes the number of problemsetters present on the room. **N + 1** lines follow the first line. Each of the next **N** lines has 2 integers, **r (0 <= r < R) & c (0 <= c < C) ,** position of a problemsetter. For your reference, the upper left corner of the cell has **(0,0)** while the lower left one has (**R-1, 0)** and the lower right one has **(R-1, C-1)** co-ordinate. The last line gives the co-ordinate of Brother Arif in the same format. The last test case will be followed by a case with **R = C = N = 0**. This case should not be processed.

Output

For each test case, print a line. This line starts like, "Case X: ", where, **X** is the case number. After that print a string based on Brother Arif's destiny. If Brother Arif can find a cell where none else can catch him, print **"Escaped again! More 2D grid problems!".** If someone can capture him no matter in which cell he goes, print "**Party time! Let's find a restaurant!"**

Sample Input	Sample Output
5 5 2 0 1 4 2 1 2 5 5 3 0 1 4 2	Case 1: Escaped again! More 2D grid problems! Case 2: Party time! Let's find a restaurant!

4 3 1 2 0 0 0	

Problem Setter: Mohammad Mahmudur Rahman, Special Thanks: Sohel Hafiz