Problem A : Lobby

You are a member of the "Conservative Party" in the parliament. There is one more party called the "Reformist Party". Other members of the parliament are independents and do not belong to either of these two parties. When a bill is put for vote, each member of the two parties has to vote as being decided by his/her party. But an independent member makes his/her mind on how to vote based on different lobbying that always exists. Note that parliament uses the majority rule (more than half of all members) to make decision.

Suppose a bill is intended to put for vote on the floor in the parliament and you know that the Reformist party is against the bill. You and your party-mates want to do lobbying on the independent members so that enough of them make or change their minds so that the bill is passed. Your task is to find the minimum number of independent members that have to vote as you wish so that the bill is passed.

Input (Standard Input)

There are multiple test cases in the input. Each test case appears in one line containing three space-separated integers n, m and k which respectively are the total number of members, the number of members in the Conservative Party and the number of members in the Reformist Party ($1 \leqslant n, m, k \leqslant 1000, m+k \leqslant n$). The input terminates with a line containing `0 0 0` which should not be processed.

Output (Standard Output)

For each test case, output a line containing the minimum number of independent members that have to vote as you wish so that the bill is passed. Output `-1`, if there is no way to pass the bill.

Sample Input and Output

Standard Input	Standard Output

8 3 3
12 4 6
7 4 3
0 0 0
	2
	-1
	0

Problem B : Cameras

In a crowded city, a network of traffic cameras are installed on all crosses and capture the image of the cars that pass the red lights. The law enforcement department requires a software system to process all the captured numbers, and issue a ticket for all the cars that violate the law. An image processing module has already preprocessed the video recordings and generated a file containing the recorded car numbers. You are requested to write the ticketing module. The problem is that the image processing module is not perfect, and some car numbers are wrong.

A true car number should meet all of the following requirements:

- It is 8 characters long.
- The two leftmost characters are identical digits between 1 to 9, which indicate in which city the car number is issued.
- The following two characters are two digits between 1 to 9.
- The following character is a capital English letter.
- The three rightmost characters are also digits between 1 to 9.

Input (Standard Input)

The first line of input contains the number of captured car numbers (n) that are reported by the image processing module. In each of the next n lines, there are exactly 8 numerical or English alphabetical characters, showing one inferred car number. You can assume $1 \leqslant n \leqslant 1000$.

Output (Standard Output)

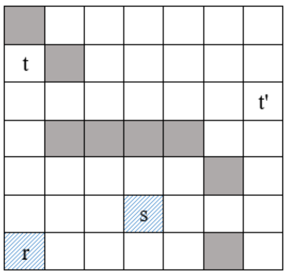
You should report the car numbers that have violated the law, one in each line, in the same order that they appear in the input. If the same car number has violated the law multiple times, all of the violation cases should be reported. The wrong car numbers, which do not meet one or more of the above requirements, should not be listed in the output.

Sample Input and Output

Standard Input	Standard Output
5 8835R551 4352S132 2241X223 55123456 9914t521	8835R551 2241X223

Problem C : Robot

We have recently acquired a robot to help us move heavy boxes around in a factory. To move a box toward a direction, the robot first needs to reach the box from behind, and then push it forward in the desired direction.



There are several obstacles inside the factory. We represent the factory by an $m \times n$ grid, and mark obstacles as "blocked" cells inside the grid. Moreover, we represent the robot and the boxes as 1×1 squares, each occupying a single cell. An example is illustrated in the figure on the right. In this figure, blocked cells are shown in gray, and the positions of the robot and a box are marked by r and s, respectively.

We call a grid cell "free" if it is not blocked, nor it is occupied by a box. At each step, the robot can move from its current position to one of its four neighboring cells, located at the left, right, top, and bottom of the current cell, provided that the neighboring cell is free. If the neighboring cell is occupied by a box, the robot can push the box forward in the same direction that the robot is moving, provided that the cell to which the box enters is free. Obviously, the robot and the boxes can never cross the boundary of the grid.

Suppose that there is only one box inside the factory, located at a start position s. Assume that we want to move the box to a target position t using the robot. We say that t is "reachable" from s if there is a sequence of moves for the robot to push the box from position s to position t. For example, in the figure above, the leftmost cell marked by t is reachable from s, but the other cell at position t' is not reachable. Given a grid containing a box at position s and a robot at position r, your task is to write a program to check how many of the grid cells are reachable from s using the robot. Note that the cell s itself is always counted as a reachable cell, even if the robot can never reach the box.

Input (Standard Input)

There are multiple test cases in the input. The first line of each test case contains two positive integers m and n ($1 \leqslant m, n \leqslant 1000$), which indicate the size of the test case grid, $m \times n$. Each of the next m lines contains a string of length n, where the jth character of the ith line represents the cell (i, j) of the grid. Obstacles are represented by character `o`. The position of the robot and the start position of the box are represented by characters `r` and `s`, respectively. Other cells are filled by `-` characters. The input terminates with a line containing `0 0` which should not be processed.

Output (Standard Output)

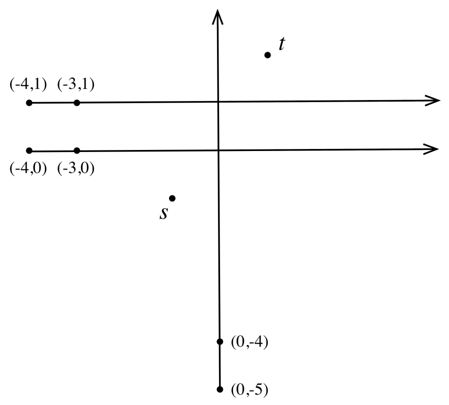
For each test case, output a line containing the number of grid cells reachable from the start position of the robot and box.

Sample Input and Output

Standard Input	Standard Output
7 7 o------ -o----- ------- -oooo-- -----o- ---s--- r----o- 3 4 ---o -os- ---r 0 0	21 6

Problem D : Laser Game

You are participating in a game with several of your classmates. Each of your opponents has a device that generates a laser beam from the location of that person in the direction that (s)he chooses all the way to the infinity. Once all of the beams are chosen and fixed, your turn starts. Your job is to run from your current location s to the destination t in a path that crosses the minimum number of the laser beams.



For simplicity, we assume that the game is played on a two-dimensional plane with your n opponents, $o_1, \ldots, o_n$ located in n distinct points $p_1, p_2, \cdots, p_n$. Each opponent o_i starts his/her one-way laser beam in a direction. Some beams can be in parallel. Two points s and t (different from other points) are given. You are standing in s and have to run in a path to reach t such that the running path crosses the minimum number of beams.

Input (Standard Input)

There are multiple test cases in the input. The first line of each test case contains n, the number of laser beams ($1 \leqslant n \leqslant 200$). Each of the next n lines contains 4 space-separated integers. The first two integers specify the x and y coordinates of the location of an opponent, and the other two integers are the x and y coordinates of a point lying on the laser beam. The last line contains the x and y coordinates of s and t, respectively. You can assume no three points of the input are co-linear and the absolute value of the coordinates will not exceed 10^8. The input terminates with a line containing `0` which should not be processed.

Output (Standard Output)

For each test case, output a line containing the minimum number of beams to cross.

Sample Input and Output

Standard Input	Standard Output

2
-1 -1 1 1
-1 1 1 -1
0 2 0 -2
6
-2 0 20 0
-2 3 20 3
6 1 -10 1
5 -5 5 20
1 -2 7 4
4 5 -4 -3
4 4 3 -2
3
-4 0 -3 0
-4 1 -3 1
0 -5 0 -4
-1 -1 1 2
0
	0
	2
	1

Problem E : Billboard

Since many years ago, "Hadoop Advertising Co." has installed giant advertisement LED billboards in the city. A billboard is a rectangular array of LED diodes, consisting of m LED rows that are connected to a central controller board. Each LED row is a thin circuit with n LED diodes that serve as the pixels.

Due to the low quality, there are many dead LED diodes on the company billboards now. You, as the chief electronics engineer of the company, are assigned to the task of repairing the billboards. The problem is that the LED billboards are quite old, and there is a limited stock of spare parts available to repair each. There are two types of spare parts: the single LED diodes that can be used to fix a single dead pixel, and the LED rows that fix a whole row of the billboard.

The strategy of the company is to show the advertisements on a sub-rectangular region of each billboard that has no dead pixels, and turning off the other parts of the billboard. Your goal is to make the largest possible area without dead pixels using the spare parts.

Input (Standard Input)

There are multiple test cases in the input. The first line of each test case contains 4 space-separated non-negative integers, the number of rows (m), the number of LED diodes in each row (n), the number of spare rows (r), and the number of spare diodes (s) available for repair. The following m lines show the current state of the rows of the billboard from top to bottom. Each line contains n digits 1 (a good pixel) or 0 (a dead pixel). You can assume both m and n are between 1 and 300 inclusive. The input terminates with a line containing `0 0 0 0` which should not be processed.

Output (Standard Output)

For each billboard you should print a single line containing the maximum number of pixels in a sub-rectangular region of the billboard containing no dead pixels after the repair.

Sample Input and Output

Standard Input	Standard Output

5 7 1 1
1 1 1 1 1 0 0
0 0 0 0 0 1 1
1 1 1 0 1 0 0
1 1 1 1 1 0 1
1 1 1 1 1 1 1
4 4 3 4
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
5 5 1 2
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
0 0 0 0
	25
	16
	10

Problem F : Funfair

We are going to a funfair where there are n games $G_1, \ldots, G_n$. We want to play k games out of the n games, and we can choose the order in which we play them—note that we cannot play any game more than once. We have to specify these k games and their order before starting any game.

At each point in time, we have some amount of money, which we use in playing the games. At the beginning, we have x_0 Oshloobs of money. If before playing game G_i, we have x Oshloobs and we win in G_i, our money increases to $x + A_i$ for some $A_i \geqslant 0$. If we have x Oshloobs before playing game G_i and we lose in G_i, we lose L_i percent of x. The probability that we win game G_i (independently of other games) is P_i percents.

The goal is to play k of the games in such an order to maximize the expected amount of money we end up with after playing all k selected games in that order.

Input (Standard Input)

There are multiple test cases in the input. The first line of each test case contains three space-separated integers n, k, and x_0 ($1 \leqslant k \leqslant n \leqslant 100$, $0 \leqslant x_0 \leqslant 10^6$). Each of the next n lines specifies the properties of game G_i with three space-separated integers A_i, L_i, and P_i ($0 \leqslant A_i, L_i, P_i \leqslant 100$). The input terminates with a line containing `0 0 0` which should not be processed.

Output (Standard Output)

For each test case, output a single line containing the maximum expected amount of our final money rounded to exactly two digits after the decimal point.

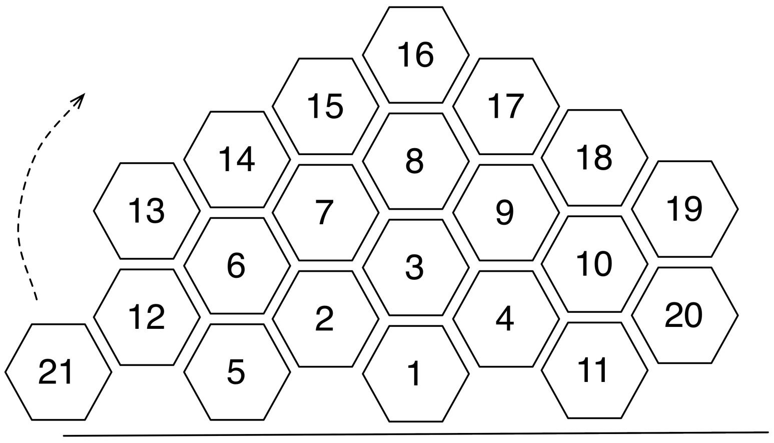
Sample Input and Output

Standard Input	Standard Output

2 2 100
10 0 50
100 10 20
2 1 100
10 0 50
100 10 20
0 0 0
	117.00
	112.00

Problem G : Beehive

There is an infinite beehive like the one given in the figure. We consider two cells to be adjacent if and only if they share a side. A path of length k from cell c_0 to cell c_k is a sequence of cells $c_0, c_1, \ldots, c_k$ such that c_i and c_{i+1} are adjacent for all $0 \leqslant i < k$. The distance between cells i and j is the length of the shortest path from cell i to cell j.



The cells of the beehive are indexed using positive integers as shown. The cells with larger distance from cell 1 are given larger indices. The indices of cells with the same distance from cell 1 increases from left to right. Each positive integer is the index of exactly one cell.

We want to know the distance of two cells whose indices are given.

Input (Standard Input)

There are multiple test cases in the input. Each test case is a single line containing two space-separated integers i and j as the indices of two cells ($1 \leqslant i, j \leqslant 10^4$). The input terminates with a line containing `0 0` which should not be processed as a test case.

Output (Standard Output)

For each test case, output a single line containing the distacne of the given cells.

Sample Input and Output

Standard Input	Standard Output

8 4
11 12
365 365
0 0
	2
	5
	0

Problem H : DNA Sequencing

Finally, Plankton's attempts to steal the Krabby Patty formula succeeded and it eventually put the Krusty Krab out of business. So, SpongeBob and his co-workers decided to switch to a brand new job. Their new startup is Krusty-Royan, a biological research institute whose main focus is on DNA sequencing. Their first customer is Sandy, the squirrel scientist, who has found the corpse of an alien from the outer space and asked Krusty-Royan crew to extract its DNA sequence. Contrary to the life on earth, the DNA of the alien was not only composed of the 4 well-known nucleotides (A, C, G, and T), but all 26 English letters! So, each part of its DNA is a sequence of capital English letters. Given the alien tissue, the DNA sequencer machine extracted a number of (not necessarily distinct) DNA sequences and printed them on paper, one per line.

Based on the contract, a DNA sequence is ***valid*** only if its length is at least M, and Sandy will pay one dollar for each ***distinct*** valid DNA sequence. So, Mr. Krabs, the greedy boss of Krusty-Royan has asked SpongeBob to use a correction pen and erase some letters from the end of the sequences printed on the paper in order to maximize the number of distinct valid DNA sequences. Your job is to help SpongeBob find the maximum number of distinct valid DNA sequences he can make.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing two space-separated integers k and M $(1 \leqslant k \leqslant 500, 1 \leqslant M \leqslant 500)$. Each of the next k lines starts with a number n_i followed by a string s_i which means there are n_i copies of DNA sequence s_i printed on the paper $(1 \leqslant n_i \leqslant 500)$. The length of the strings is a positive integer not greater than 500. The input terminates with a line containing `0 0` which should not be processed as a test case.

Output (Standard Output)

For each test case, output a line containing the maximum number of distinct valid DNA sequences which SpongeBob can provide.

Sample Input and Output

Standard Input	Standard Output
2 1 2 ABB 2 ABC 2 2 2 ABB 2 ABC 2 3 2 ABB 2 ABC 2 4 2 ABB 2 ABC 0 0	4 3 2 0

Problem I : Cafebazaar

You are working as the product manager of *Cafebazaar*, a well-known Iranian Android marketplace. As your one-year plan, you have a list of applications to develop for which you need to assign proper developers. There are n developers in Cafebazaar, some of which are full-time, and the others are part-time developers. Your have m applications in your list, some of which are business-critical and the others are ordinary applications.

Of course, developers have different skills. Some developers are professional in developing some applications, while they are incapable of developing the other applications. If developer i is capable of developing application j, then you will gain $x_{i,j}$ amount of payoff by assigning her to that application in your plan. Obviously, a developer cannot be assigned to an application that she is not capable to develop. To gain maximum time and cost efficiency, you would like to assign each developer to at most one application, and assign at most one developer to each application. A development plan is *proper* if it has the following two properties: (i) all full-time developers are involved in the plan, so they will not get disappointed, and (ii) all critical applications are covered, hence you will not lose clients. In other words, a development plan is proper if each full-time developer is assigned to exactly one application, and exactly one developer is assigned to each critical application. Note that in a proper plan, you may also assign part-time developers to applications, or assign developers to ordinary applications. Your goal is to find a proper plan that yields a maximum amount of payoff.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing two space-separated integers n and m ($1 \leqslant n, m \leqslant 100$), representing the number of developers and applications, respectively. The next line starts with an integer t ($0 \leqslant t \leqslant n$), the number of full-time developers, followed by t numbers, each of which is the index of a full-time developer. Developers are indexed from 1 to n. The next line starts with an integer s ($0 \leqslant s \leqslant m$), the number of critical applications, followed by s numbers, each of which is the index of a critical application. Applications are indexed from 1 to m. The next n lines, one for each developer, contain the following information. The i-th line ($1 \leqslant i \leqslant n$) starts with an integer d_i ($0 \leqslant d_i \leqslant m$), the number of applications that developer i is capable to develop, followed by d_i pairs of integers $a_{i,j}$ and $x_{i,j}$ ($1 \leqslant j \leqslant d_i, 1 \leqslant a_{i,j} \leqslant m, 1 \leqslant x_{i,j} \leqslant 10^6$), where $a_{i,j}$ is the index of an application that developer i is capable to develop, and $x_{i,j}$ is the payoff achieved if application $a_{i,j}$ is developed by developer i. Note that each $a_{i,j}$ as the index of an applicaton appears at most once in this list of a developer. The input terminates with a line containing `0 0` which should not be processed.

Output (Standard Output)

You should output one line for each test case, containing the maximum payoff you can obtain by a proper plan. If there is no proper plan, you must output -1 in one line.

Sample Input and Output

Standard Input	Standard Output

2 4
1 1
1 3
2 1 8 2 10
3 2 2 3 10 4 50
4 3
3 1 2 4
2 1 3
1 1 200
2 2 700 3 200
2 2 300 3 100
1 1 500
0 0
	20
	-1

Problem J : Fence

M-O (Microbe-Obliterator) is a tiny robot, programmed to clean any microbes it detects in the Axiom, a starliner spacecraft built by the Buy n Large (BNL) corporation. Tonight M-O is too busy since the great hall of the Axiom is a mess. A number of foreign microbes have attacked the spacecraft and are spread on the floor of the great hall. It's M-O's duty to wipe all those microbes.

The floor of the great hall is an infinite grid of 1×1 cells. The 4 edges and the 2 diagonals of each cell are drawn as white segments. The microbes are currently located only on the corners of the cells.

Since the cleaning job might take a lot of time, M-O needs to build a protector fence around the microbes as soon as possible, to keep the Axiom residents safe and to prevent the microbes from spreading to the other areas of the spacecraft. Since M-O is very regular, it makes the fence as a single closed region with borders only on the white segments (cell edges and diagonals). All microbes should be located inside the fence, or on its borders.

Given the map of the microbes, your task is to help M-O build a fence with the minimum perimeter.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing a single integer n, the number of microbes in the great hall ($0 \leqslant n \leqslant 10000$). Each of the next n lines contains two integers x_i and y_i, denoting the coordinates of the i-th microbe. The absolute value of all coordinates is guaranteed to be at most 10^6. The coordinates are reported based on some arbitrary grid cell corner, taken as the coordinates origin. The input terminates with a line containing a single zero, which should not be considered as a test case.

Output (Standard Output)

It is clear that the perimeter length of any fence built over the edges and diagonals of the grid can be uniquely written as $a + b\sqrt{2}$ where a and b are two non-negative integer numbers. For each test case, write a single line containing the two integers a and b with the assumption that the perimeter length of the fence with the minimum perimeter is $a + b\sqrt{2}$. Note that a fence must be seen as a closed path; therefore if two parts of the fence overlap, the overlapped part must be taken into account twice.

Sample Input and Output

Standard Input	Standard Output

11
2 2
3 2
3 0
7 1
3 5
1 4
7 4
5 4
0 2
7 2
2 0
2
1 3
3 1
0
	12 6
	0 4