Important notes

- Problems X and Y are given out at 9:00.
- Problems A–I are given out at 10:00.
- Your program has to read the problem input from the standard input and write the solution to the standard output.
- Your program has to return 0 as return code.
- End of line is denoted by a single character of code 10, both in the input and the output.
- The last line of the input is terminated by a new line character and you have to output a new line character after the last line of the output.
- The official clock of the contest is the clock of the server. You can see the current time on the submission server webpages. (Do not forget to refresh to see the current time.)
- Due to some possible performance issues, we do not recommend using the `iostream` library in C++.

Practice Problem X: Words

Introduction

If you want to impress people by showing how educated and intelligent you are, then you should use long words such as "antidisestablishmentarianism" and "floccinaucinihilipilification." Furthermore, if you want to asses how educated and intelligent other people are, then you need a quick way of finding the longest word in a text. Your task is to write a program that determines the length of the longest word.

Input

The input contains several blocks of test cases. Each case begins with a line containing an integer $1 \leq n \leq 1000$, the length of the text in lines. The next n lines contain words separated by one or more space characters. The words contain only the letters `'a'`–`'z'`. The length of each line is at most 1000 characters.

The input is terminated by a block with $n = 0$.

Output

For each test case, you have to output a single integer on a separate line: the length of the longest word in the text.

Sample Input

```
2
a ab abc
  abcd  abcde
1
i wanna floccinaucinihilipilificate baby
0
```

Sample Output

```
5
27
```

Practice Problem Y: Divide and Rule

Introduction

Two brothers are fighting over the family legacy. Their family is very poor, so all they inherited is a large sack of rice. Your task is tell them how to divide this bag into two equal parts. The bag has to be divided as evenly as possible. If the number of grains is odd, then the first brother receives one grain more than the second.

Input

The input contains several blocks of test cases. Each case is described by a single line containing an integer $20 \leq n \leq 2000000$, the number of grains in the bag.

The input is terminated by a block with $n = 0$.

Output

For each test case, you have to output a single line containing two integers, the number of grains received by the two brothers.

Sample Input

```
100
101
0
```

Sample Output

```
50 50
51 50
```

Problem A: False Coin

Introduction

The "Gold Bar" bank received information from reliable sources that in their last group of N coins exactly one coin is false and differs in weight from other coins (while all other coins are equal in weight). After the economic crisis they have only a simple balance (with two pans) available. Using this balance, one is able to determine if the weight of objects in the left pan is less than, greater than, or equal to the weight of objects in the right pan. In order to detect the false coin the bank employees numbered all coins by the integers from 1 to N, thus assigning each coin a unique integer identifier. After that they began to weight various groups of coins by placing equal numbers of coins in the left pan and in the right pan. The identifiers of coins and the results of the weightings were carefully recorded. You are to write a program that will help the bank employees to determine the identifier of the false coin using the results of these weightings.

Input

The input contains several blocks of test cases. The first line of each test case contains two integers N and K, separated by a space, where N is the number of coins ($2 \leq N \leq 100$) and K is the number of weightings performed ($1 \leq K \leq 100$). The following $2K$ lines describe all weightings. Two consecutive lines describe each weighting. The first of them starts with a number P_i ($1 \leq P_i \leq N/2$), representing the number of coins placed in the left and in the right pans, followed by P_i identifiers of coins placed in the left pan and P_i identifiers of coins placed in the right pan. All numbers are separated by spaces. The second line contains one of the following characters: `'<'`, `'>'`, or `'='` (without quotes). It represents the result of the weighting:

- `'<'` means that the weight of coins in the left pan is less than the weight of coins in the right pan,
- `'>'` means that the weight of coins in the left pan is greater than the weight of coins in the right pan,
- `'='` means that the weight of coins in the left pan is equal to the weight of coins in the right pan.

The input is terminated by a block with $N = K = 0$.

Output

For each test case, you have to output a single integer on a separate line: the identifier of the false coin or 0, if it cannot be found by the results of the given weightings.

Sample input

```
5 3
2 1 2 3 4
<
1 1 4
=
1 2 5
=
6 4
3 1 2 3 4 5 6
<
1 1 2
=
2 1 3 4 5
<
2 4 5 2 6
>
0 0
```

Sample output

```
3
0
```

Problem B: Fuse

Introduction

Maybe you are familiar with the following situation. You have plugged in a lot of electrical devices, such as toasters, refrigerators, microwave ovens, computers, stereos, etc, and have them all running. But at the moment when you turn on the TV, the fuse blows, since the power drawn from all the machines is greater than the capacity of the fuse. Of course this is a great safety feature, avoiding that houses burn down too often due to fires ignited by overheating wires. But it is also annoying to walk down to the basement (or some other inconvenient place) to replace the fuse or switch it back on.

What one would like to have is a program that checks *before* turning on an electrical device whether the combined power drawn by all running devices exceeds the fuses capacity (and it blows), or whether it is safe to turn it on.

Input

The input consists of several test cases. Each test case describes a set of electrical devices and gives a sequence of turn on/off operations for these devices.

The first line of each test case contains three integers n, m, and c, where n is the number of devices ($n \leq 20$), m is the number of operations performed on these devices and c is the capacity of the fuse (in Amperes). The following n lines contain one positive integer c_i each, the consumption (in Amperes) of the i-th device.

This is followed by m lines also containing one integer each, between 1 and n inclusive. They describe a sequence of turn on/turn off operations performed on the devices. For every number, the state of that particular devices is toggled, i.e. if it is currently running, it is turned off, and if it is currently turned off, it will by switched on. At the beginning all devices are turned off.

The input will be terminated by a test case starting with $n = m = c = 0$.

Output

For each test case, first output whether the fuse was blown (`'YES'`) or not (`'NO'`) during the operation sequence. The fuse will be blown if the sum of the power consumptions c_i of turned on devices at some point exceeds the capacity of the fuse c. If the fuse is not blown, output on the next line the maximal power consumption (in Amperes) by turned on devices that occurred during the sequence.

Sample Input

```
2 2 10
5
7
1
2
3 6 10
2
5
7
2
1
2
3
1
3
0 0 0
```

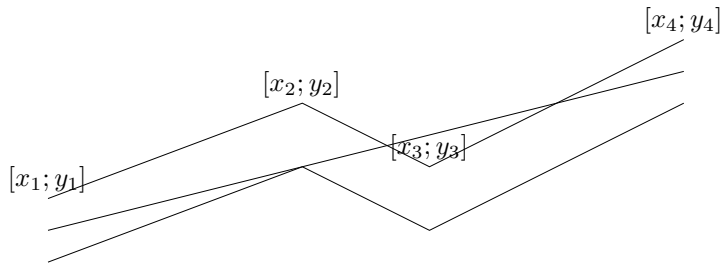
Sample Output

```
YES
NO
9
```

Problem C: Pipe

Introduction

The GX Light Pipeline Company started to prepare bent pipes for the new transgalactic light pipeline. During the design phase of the new pipe shape the company ran into the problem of determining how far the light can reach inside each component of the pipe. Note that the material which the pipe is made from is not transparent and not light reflecting.



Each pipe component consists of many straight pipes connected tightly together. For the programming purposes, the company developed the description of each component as a sequence of points $[x_1; y_1]$, $[x_2; y_2]$, ..., $[x_n; y_n]$, where $x_1 < x_2 < \cdots < x_n$. These are the upper points of the pipe contour. The bottom points of the pipe contour consist of points with y-coordinate decreased by 1. To each upper point $[x_i; y_i]$ there is a corresponding bottom point $[x_i; (y_i) - 1]$ (see picture above). The company wants to find, for each pipe component, the point with maximal x-coordinate that the light will reach. The light is emitted by a segment source with endpoints $[x_1; (y_1) - 1]$ and $[x_1; y_1]$ (endpoints are emitting light too). Assume that the light is not bent at the pipe bent points and the bent points do not stop the light beam.

Input

The input file contains several blocks each describing one pipe component. Each block starts with the number of bent points $2 \leq n \leq 20$ on a separate line. Each of the next n lines contains a pair of real values x_i, y_i separated by space. The last block is denoted with $n = 0$.

Output

The output file contains lines corresponding to blocks in input file. To each block in the input file there is one line in the output file. Each such line contains either a real value, written with precision of two decimal places, or the message `'Through all the pipe.'` (without quotes). The real value is the desired maximal x-coordinate of the point where the light can reach from the source for corresponding pipe component. If this value equals to x_n, then the message `'Through all the pipe.'` will appear in the output file.

Sample input

```
4
0 1
2 2
4 1
6 4
6
0 1
2 -0.6
5 -4.45
7 -5.57
12 -10.8
17 -16.55
0
```

Sample input

```
4.67
Through all the pipe.
```

Problem D: Robot

Introduction

The Robot Moving Institute is using a robot in their local store to transport different items. Of course the robot should spend only the minimum time necessary when travelling from one place in the store to another. The robot can move only along a straight line (track). All tracks form a rectangular grid. Neighbouring tracks are one meter apart. The store is a rectangle $N \times M$ meters and it is entirely covered by this grid. The distance of the track closest to the side of the store is exactly one meter. The robot has a circular shape with diameter equal to 1.6 meter. The track goes through the center of the robot. The robot always faces north, south, west or east. The tracks are in the south–north and in the west–east directions. The robot can move only in the direction it faces. The direction in which it faces can be changed at each track crossing. Initially the robot stands at a track crossing. The obstacles in the store are formed from pieces occupying $1m \times 1m$ on the ground. Each obstacle is within a 1×1 square formed by the tracks. The movement of the robot is controlled by two commands. These commands are `GO` and `TURN`.

The `GO` command has one integer parameter $n \in \{1, 2, 3\}$. After receiving this command the robot moves n meters in the direction it faces.

The `TURN` command has one parameter which is either `left` or `right`. After receiving this command the robot changes its orientation by 90 degrees in the direction indicated by the parameter.

The execution of each command lasts one second.

Help researchers of RMI to write a program which will determine the minimal time in which the robot can move from a given starting point to a given destination.

Input

The input contains several blocks of test cases. The first line of each block contains two integers $M \leq 50$ and $N \leq 50$ separated by one space. In each of the next M lines there are N numbers one or zero separated by one space. One represents obstacles and zero represents empty squares. (The tracks are between the squares.) The block is terminated by a line containing four positive integers $B_1 B_2 E_1 E_2$ each followed by one space and the word indicating the orientation of the robot at the starting point. B_1, B_2 are the coordinates of the square in the north–west corner of which the robot is placed (starting point). E_1, E_2 are the coordinates of square to the north–west corner of which the robot should move (destination point). The orientation of the robot when it has reached the destination point is not prescribed. We use (row, column)–type coordinates, i.e. the coordinates of the upper left (the most northwest) square in the store are 0,0 and the lower right (the most southeast) square are $M - 1, N - 1$. The orientation is given by the words `north` or `west` or `south` or `east`. The input last block contains only one line with $N = 0$ and $M = 0$.

Output

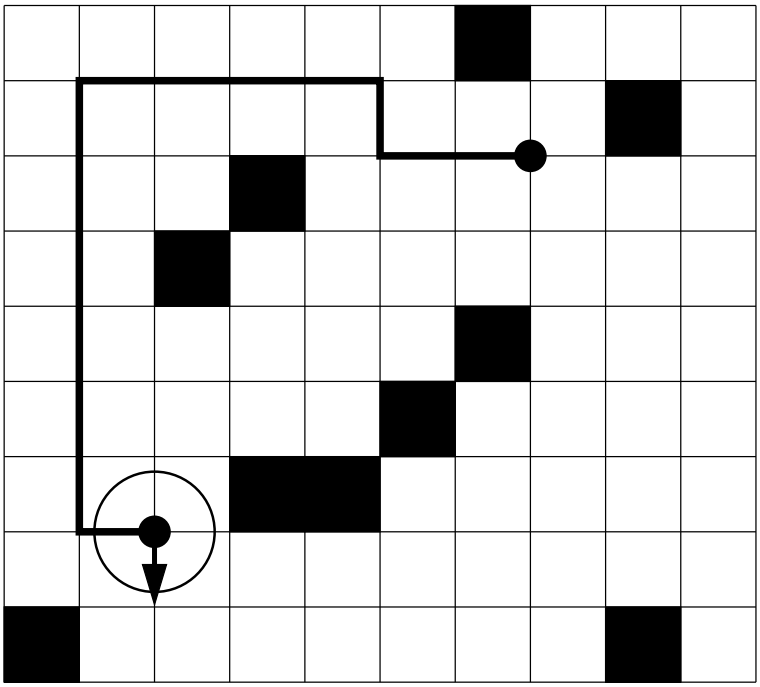
For each test case, you have to output a single line containing the minimal number of seconds in which the robot can reach the destination point from the starting point. If there does not exist any path from the starting point to the destination point the line will contain -1.

Sample Input

```
9 10
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 1 0
7 2 2 7 south
0 0
```

Sample Output

```
12
```



Problem E: L-system

Introduction

A D0L (deterministic Lindenmayer system without interaction) system consists of a finite set Σ of symbols (the alphabet), a finite set P of productions and a starting string ω. The productions in P are of the form $x \to u$, where $x \in \Sigma$ and $u \in \Sigma^+$ (u is called the right side of the production), Σ^+ is the set of all strings of symbols from Σ excluding the empty string. Such productions represent the transformation of the symbol x into the string u. For each symbol $x \in \Sigma$, P contains exactly one production of the form $x \to u$. Direct derivation from string u_1 to u_2 consists of replacing each occurrence of the symbol $x \in \Sigma$ in u_1 by the string on the right side of the production for that symbol. The language of the D0L system consists of all strings which can be derived from the starting string ω by a sequence of the direct derivations.

Suppose that the alphabet consists of two symbols `a` and `b`. So the set of productions includes two productions of the form $\texttt{a} \to u$, $\texttt{b} \to v$, where u and $v \in \{\texttt{a}, \texttt{b}\}^+$, and the starting string $\omega \in \{\texttt{a}, \texttt{b}\}^+$. Can you answer whether there exists a string in the language of the D0L system of the form xzy for a given string z? (x and y are some strings from Σ^*, where Σ^* is the set of all strings of symbols from Σ, including the empty string.). Certainly you can. Write the program which will solve this problem.

Input

The input consists of several blocks of lines. Each block includes four lines. There are no empty lines between any successive two blocks. The first line of a block contains the right side of the production for the symbol `a`. The second one contains the right side of the production for the symbol `b` and the third one contains the starting string ω and the fourth line the given string z. The right sides of the productions, the given string z and the starting string ω are at most 15 characters long.

The input is terminated with a line containing 0.

Output

For each block in the input there is one line in the output containing `YES` or `NO` according to the solution of the given problem.

Sample Input

```
aa
bb
ab
aaabb
a
b
ab
ba
0
```

Sample Output

```
YES
NO
```

Problem F: Obfuscation

Introduction

It is a well-known fact that if you mix up the letters of a word, while leaving the first and last letters in their places, words still remain readable. For example, the sentence "tihs snetncee mkaes prfecet sesne", makes perfect sense to most people.

If you remove all spaces from a sentence, it still remains perfectly readable, see for example: "thissentencemakesperfectsense", however if you combine these two things, first shuffling, then removing spaces, things get hard. The following sentence is harder to decipher: "tihssnetnceemkaesprfecetsesne".

You're given a sentence in the last form, together with a dictionary of valid words and are asked to decipher the text.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with a string s: the sentence to decipher. The sentence consists of lowercase letters and has a length of at least 1 and at most 1 000 characters.
- One line with an integer n with $1 \leq n \leq 10000$: the number of words in the dictionary.
- n lines with one word each. A word consists of lowercase letters and has a length of at least 1 and at most 100 characters. All the words are unique.

Output

Per testcase: One line with the deciphered sentence, if it is possible to uniquely decipher it. Otherwise `impossible` or `ambiguous`, depending on which is the case.

Sample input

```
3
tihssnetnceemkaesprfecetsesne
5
makes
perfect
sense
sentence
this
hitehre
2
there
hello
hitehre
3
hi
there
three
```

Sample input

```
this sentence makes perfect sense
impossible
ambiguous
```

Problem G: Summits

Introduction

You recently started working for the largest map drawing company in Tibet. Part of your job is to determine what the summits in a particular landscape are. Unfortunately, it is not so easy to determine which points are summits and which are not, because we do not want to call a small hump a summit.

For example look at the landscape given by the sample input. We call the points of height 3 summits, since there are no higher points. But although the points of height 2, which are to the left of the summit of height 3, are all higher than or equal to their immediate neighbours, we do not want to call them summits, because we can reach a higher point from them without going to low (the summits of height 3). In contrast, we do want to call the area of height 2 on the right a summit, since if we would want to walk to the summit of height 3, we first have to descend to a point with height 0.

After the above example, we introduce the concept of a d-summit. A point, with height h, is a d-summit if and only if it is impossible to reach a higher point without going through an area with height smaller than or equal to $h - d$.

The problem is, given a rectangular grid of integer heights and an integer d, to find the number of d-summits.

Input

The input consists of several test cases. Each test case consist of the following:

- One line with three integers $1 \leq h \leq 500$, $1 \leq w \leq 500$ and $1 \leq d \leq 1000000000$. h and w are the dimensions of the map. d is as defined in the text.
- h lines with w integers, where the x-th integer on the y-th line denotes the height $0 \leq h \leq 1000000000$ of the point (x, y).

The input is terminated with a line $k = w = d = 0$.

Output

Per testcase: One line with the number of summits.

Sample Input

```
6 10 2
0 0 0 0 0 0 0 0 0 0
0 1 2 1 1 1 1 0 1 0
0 2 1 2 1 3 1 0 0 0
0 1 2 1 3 3 1 1 0 0
0 2 1 2 1 1 1 0 2 0
0 0 0 0 0 0 0 0 0 0
0 0 0
```

Sample Output

```
4
```

Problem H: Escape from Enemy Teritory

Introduction

A small group of commandos has infiltrated deep into enemy territory. They have just accomplished their mission and now have to return to their rendezvous point. Of course they don't want to get caught even if the mission is already over. Therefore they decide to take the route that will keep them as far away from any enemy base as possible.

Being well prepared for the mission, they have a detailed map of the area which marks all (known) enemy bases, their current position and the rendezvous point. For simplicity, we view the the map as a rectangular grid with integer coordinates (x, y) where $0 \leq x < X$, $0 \leq y < Y$. Furthermore, we approximate movements as horizontal and vertical steps on this grid, so we use Manhattan distance: $\text{dist}((x_1, y_1), (x_2, y_2)) = |x_2 - x_1| + |y_2 - y_1|$. The distance from enemy bases are also calculated using this distance function. The commandos can only travel in vertical and horizontal directions at each step.

Can you help them find the best route? Of course, in case that there are multiple routes that keep the same minimum distance to enemy bases, the commandos want to take a shortest route that does so. Furthermore, they don't want to take a route off their map as it could take them in unknown, dangerous areas, but you don't have to worry about unknown enemy bases off the map.

Input

The input contains several blocks of test cases. Each test case consists of the following:

- One line with three positive numbers N, X, Y. $1 \leq N \leq 10000$ is the number of enemy bases and $1 \leq X, Y \leq 1000$ is the size of the map: coordinates x, y are on the map if $0 \leq x < X$, $0 \leq y < Y$.
- One line containing two pairs of coordinates x_i, y_i and x_r, y_r: the initial position of the commandos and the rendezvous point.
- N lines each containing one pair of coordinates x, y of an enemy base.

All pairs of coordinates are on the map and different from each other. The input is terminated by a block with $N = X = Y = 0$.

Output

Per testcase: One line with two numbers separated by one space: the minimum separation from an enemy base and the length of the route.

Sample input

```
1 2 2
0 0 1 1
0 1
2 5 6
0 0 4 0
2 1
2 3
0 0 0
```

Sample input

```
1 2
2 14
```

Problem I: Assemble

Introduction

Recently your team noticed that the computer you use to practice for programming contests is not good enough anymore. Therefore, you decide to buy a new computer. To make the ideal computer for your needs, you decide to buy separate components and assemble the computer yourself. You need to buy exactly one of each type of component. The problem is which components to buy. As you all know, the quality of a computer is equal to the quality of its weakest component. Therefore, you want to maximize the quality of the component with the lowest quality, while not exceeding your budget.

Input

The input contains several blocks of test cases. Each test case consists of the following:

- One line with two integers: $1 \leq n \leq 1000$, the number of available components and $1 \leq b \leq 1000000000$, your budget.
- n lines in the following format: `'type name price quality'`, where `type` is a string with the type of the component, `name` is a string with the unique name of the component, `price` is an integer ($0 \leq$ `price` ≤ 1000000) which represents the price of the component and `quality` is an integer ($0 \leq$ `quality` ≤ 1000000000) which represents the quality of the component (higher is better). The strings contain only letters, digits and underscores and have a maximal length of 20 characters.

It will be always possible to construct a computer with your budget. The input is terminated by a block with $n = b = 0$.

Output

For each test case in the input, you have to output a single line containing one integer, the maximal possible quality.

Sample input

```
18 800
processor 3500_MHz 66 5
processor 4200_MHz 103 7
processor 5000_MHz 156 9
processor 6000_MHz 219 12
memory 1_GB 35 3
memory 2_GB 88 6
memory 4_GB 170 12
mainbord all_onboard 52 10
harddisk 250_GB 54 10
harddisk 500_FB 99 12
casing midi 36 10
monitor 17_inch 157 5
monitor 19_inch 175 7
monitor 20_inch 210 9
monitor 22_inch 293 12
mouse cordless_optical 18 12
mouse microsoft 30 9
keyboard office 4 10
0 0
```

Sample input

```
9
```