Problem A. Airlines

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

There are M airports and K airline companies in a country. The K companies operate a total of N direct flights between the M airports. Naturally, all the flights carry passengers in both directions, and every flight is operated by one company only. There may be several direct flights between any two airports, but each company runs at most one of these. For every pair of airports a passenger can fly from one to another using either a direct flight or through several intermediate airports.

An officer of the aviation administration has to evaluate the quality of service of the airlines. For the evaluation, the officer has to fly as a passenger once on each of the N flights (in one direction only). To avoid any suspicion of cooperation between the evaluator and the airlines, the evaluator is not allowed to fly on two flights of the same company one after another. The first and the last flights should be operated by different companies too.

The officer wants to eliminate any unnecessary expenses and plan the route so that:

- the evaluation route starts from the airport 1;
- each subsequent flight starts in the same airport where the previous one finishes;
- the evaluation route finishes at the airport 1.

Write a program to help him plan the route!

Input

The first line of the input file contains three integers M, N, and K ($2 \le M \le 1000$; $1 \le N \le 10^4$; $1 \le K \le 10$). Each of the following N lines describes one flight. Each line contains three integers: the numbers of the airports the flight connects and the number of the company that operates the flight. The airports are numbered from 1 to M and the companies from 1 to K. The flights are numbered from 1 to N in the order in which they appear in the input file.

Output

If there is no solution for some reason, output the word “`No`” on the first and only line of the file. If there is a solution, the first line of the file should contain the word “`Yes`’, and each of the following N lines should contain the numbers of the flights in the order in which the evaluator tests them. If there are several solutions, output any one of them.

Example

standard input	standard output

4 7 3
3 2 1
1 2 2
3 4 1
3 4 2
3 4 3
1 2 3
2 4 2
	Yes
	2
	1
	5
	4
	3
	7
	6

4 7 2
3 2 1
1 2 1
3 4 1
3 1 2
3 1 1
4 2 1
4 2 2
	No

Problem B. Battle Robots

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	512 mebibytes

The scientists are developing a new type of highly intelligent battle robots. It's assumed that these robots, when completed, will be able to parachute in a certain region of the Earth (You can assume that The Earth in this task is flat) and to complete special secret missions while coordinating their actions.

The develoment of the robots advances rapidly. The robots are so intellectual now that any of them can choose any arbitrary direction and start a linear movement towards that direction. But, there's still one problem: the robots, as for now, weren't taught about how to stop. However, it's not actually a problem right now, because during the development process, the robots always can be placed inside a room with soft walls, and they will stop automatically after they collide with a wall.

The main task now is to teach the robots to gather in one point. The senior programmer has already written some algorithm which chooses robots' movement direction. It's now the time to test the code which has already been written.

n robots are selected. At the initial moment of time, all the robots are placed inside a rectangular room. Their initial coordinates are known and equal to (x_i, y_i). After that, all the robots start moving simultaneously, each in its own direction. These directions are known and equal to (vx_i, vy_i).

If a robot collides into a wall, it stops its movement immediately, and before this happens, the robots' coordinates at any nonnegative real moment of time t equal to $(x_i + t \cdot vx_i, y_i + t \cdot vy_i)$. If several (two or more) robots happen to appear in one point simultaneously, nothing happens (they don't collide), and each of these robots continues its movement in its original direction.

Thereby, at any nonnegative moment of time the coordinates of all robots are known. So, the convex hull of the robots' points can be calculated at any moment of time. Let's define the quality of the algorithm as the minimal over all moments of time area of that convex hull.

Your task is to find the (nonnegative) moment of time t such that the area of the described convex hull at this moment is minimal, and calculate the convex hull's area at this moment.

Input

The first line of the input contains an integer number n: the number of robots ($3 \le n \le 80$). The second line contains two integer numbers w and h separated by whitespace: the dimensions of the room ($2 \le w, h \le 10^4$). The room is a rectangle with its sides parallel to coordinate axes, and with vertices at points $(0, 0)$, $(w, 0)$, $(0, h)$ and (w, h). Each of the following n lines contains four integer numbers x_i, y_i, vx_i and vy_i separated by whitespace: coordinates and speed of the corresponding robot ($0 < x < w$, $0 < y < h$, $0 \le |vx|, |vy| \le 100$).

Output

Your program must print the moment of time at which the convex hull's area is minimal on the first line. On the second line, print that area. If there are several such moments of time, you are allowed to choose any of them. Both numbers must be real and must lie inside the interval from 0 to 10^9. The answer will be assumed correct if the absolute or relative error of both the numbers doesn't exceed 10^{-6}.

Examples

standard input	standard output

3
10 10
1 1 0 2
8 8 -2 0
8 6 0 1
	3.5
	0.0

3
10 10
1 1 -1 3
9 1 1 3
5 3 0 2
	1.0
	5.0

3
10 10
1 1 0 0
1 2 0 0
2 1 0 0
	0.0
	0.5

Problem C. Circles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

This is an interactive problem.

There is a square with n non-degenerate circles ($1 \le n \le 2\,000$) wholly inside it. The left bottom corner of the square is $(0, 0)$ and upper right one is $(20, 20)$. Both centre coordinates and radiuses of the circles are integer numbers.

You can only query the number of circles whose boundary or internals contain the point specified by you. Your task is to find n.

Interaction Protocol

Each plain query consists of two real numbers, all separated by single space — coordinates of some point inside the square. The response will be a single integer — the number of circles containing given point.

Do not forget to print end-of-line symbol and `flush()` output after each query!

Your program should indicate the answer when ready using the special word `Done` and an integer — answer to the problem.

The number of queries shouldn't exceed 5 000.

Output

Output line containing either two real numbers x and y ($0 \le x, y \le 20$) or word "`Done`" and integer number (the answer to the problem) separated by space.

Note that all real numbers will be rounded in such a way that their fractional part will have at most 5 digits.

Input

Input contains lines with integers — answers to your queries.

Example

standard input	standard output
`2`	`1.0 1.0`
`2`	`1.0 0.0`
`2`	`0.0 1.0`
`2`	`2.0 1.0`
`2`	`1.0 2.0`
`1`	`5.0 5.0`
`1`	`5.0 7.0`
`1`	`5.1 6.0`
	`Done 3`

Here two identical circles have its centers in $(1, 1)$ and radii 1. Third one has centre in $(5, 6)$ and radius 2.

Problem D. Davy Jones Tentacles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Captain Davy Jones keeps his treasure in a chest. The lock on the chest is a checkered rectangular panel size of $h \times w$ cells. There is one key in each cell of the panel. The combination lock opens if you click on certain keys in a specific order.

In order not to waste time, the sequence of clicks should be processed as quickly as possible, so Davy Jones uses his k tentacles to open the chest. Initially, all the tentacles are located above the left top key.

Tentacles of Davy Jones exist in four-dimensional space, so we may assume that they are able to move independently. During one second, a tentacle can either stay still, move to the position above one of the eight neighboring keys, or press the key which is located below it. One press takes a whole second, and during this second, the tentacle pressing the key can not move, and other tentacles can not press other keys.

Write a program that, given a sequence of n keystrokes required to open the chest, finds the minimum time in which the chest can be opened, and also provides a plan: which tentacles at which moments should produce these keystrokes.

Input

The first line of the input file contains four integers h, w, k and n: height and width of the panel, number of tentacles and length of the requied sequence of keystrokes ($2 \le h, w \le 10$, $2 \le k \le 10$ and $1 \le n \le 1500$). The following n lines describe the desired sequence of keystrokes. Each line contains two integers r_i and c_i separated by a space: the numbers of row and column number where the key is located ($1 \le r_i \le h$, $1 \le c_i \le w$).

Output

On the first line print one integer: the minimum total time t in seconds during which you can produce all the keystrokes. Then output n lines. At the i-th of these lines, print two integers s_i and t_i: the number of the tentacle which presses i-th key in the sequence and the number of the second in which it happens ($1 \le s_i \le k$). Seconds numeration is 1-based. Remember that moments of time t_i must be strictly increasing. If there are several optimal answers, output any one of them.

Examples

standard input	standard output

3 3 2 4
1 1
2 2
3 3
1 1
	5
	1 1
	2 2
	2 4
	1 5

3 4 2 4
3 3
1 4
3 2
1 2
	7
	2 3
	1 4
	2 6
	1 7

Note

In the first example, one of the right scenarios is the following. At first second, the first tentacle presses the key at the cell $(1, 1)$. Meanwhile, the second tentacle moves to the cell $(2, 2)$, and performs a keystroke there at second second. After that, the third second is spent on movement of the second tentacle from $(2, 2)$ to $(3, 3)$, and the fourth one is spent on pressing the key in that cell. Meanwhile, the first tentacle waits in the cell $(1, 1)$ to press the key in this cell again after the third keystroke at fifth second.

In the second example, one of the right scenarios is the following. The second tentacle moves to the cell $(3, 3)$ to press the key in this cell at the third second. The first tentacle moves to the cell $(1, 4)$ to press the key in this cell at the fourth second. After that, to reach the cell $(1, 2)$ and to make a keystroke there, the first tentacle needs three additional seconds. The second tentacle can reach $(3, 2)$ by the start of the fifth second and press the key at fifth or sixth second.

Problem E. Exploring the Space

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 Mebibytes

Scientists on galaxy far far away invented N-dimensional space explorer, so-called SB (Space Boomerang). Naturally, they want it to come back to the same location. To program it, they have a bunch of modules that they can install into it. Each module gives it power along a specific direction. For each module they install, they enter a non-zero distance that the SB will move along that direction (obviously, they do not have to install all of them in one game). Distance can be both positive and negative, where negative distance means that it travels in the opposite direction. After programming, they throw the SB. It travels thorugh the space in a special way (it moves using modules simultaneously, until they are all out of power). Of course, scientists repeat start many times.

Modules can be reused. Scientests would like to check in SB more different modules. However, some of modules might be useless, because it not possible to use them and make SB coming back to the same location.

Your task is to find all "useless" modules.

Input

The first line of the input contains two integers M and N separated by a space character ($1 \le M \le 3000$, $1 \le N \le 300$). M is the number of modules, and N is the number of dimensions of the hyperspace. Each of the next M lines contain N real numbers separated by a space character that represent coordinates of a direction in N-dimensional space for the coresponding module. Modules are indexed by numbers 1 through M, in the order given in the input. Each coordinate is given with at most two decimal digits, and its absolute value is not greater than 10^5.

Output

In the first line of the output write one integer K, which is the number of "useless" modules. In the next K lines write indices of those modules, one per line, in the same order as they appear in the input. Terminate each line with a newline character.

Example

standard input	standard output

4 3
0 0 1
1 0 2
0 1 0
1 0 1
	1
	3

Problem F. Funny Graph

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Vasya calls the graph *funny*, if difference of degrees of any two its vertices does not exceed 1 by absolute value. Vasya already got some funny graph, and he wants remove some edges from it, so resulting graph will be funny and degree of at least one vertice of the new graph will be equal to d.

Help Vasya to build such a graph.

Input

First line of input file contains three integers: N $(1 \le N \le 300)$ — number of vertices of a given graph, M $(1 \le M \le N \cdot (N-1)/2)$ — number of its edges and d $(0 \le d \le 299)$ — required degree of a vertice. Each of next M lines contains numbers of vertices, connected by a edge — two distinct positive integers, each not greater than N. It is guaranteed that graph does not contain multiple edges.

Output

If Vasya cannot build such a graph by some reason, output single line containing one word "`NO`". Otherwise print "`YES`" in first line, number of edges in resulting graph in second. Third line must contain list of edges in increasing order (edges are numbered by sequential integers, stating from 1, as they appear in the input file). If there are multiple solutions, print any of them.

Examples

standard input	standard output

5 6 1
1 2
2 3
1 3
1 4
4 5
3 5
	YES
	2
	2 5

4 3 3
1 2
2 3
3 4
	NO

Problem G. Guards of Black and White

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 Mebibytes

The Kingdom of Black and White consists of n cities, connected by m bidirectional roads. Cities are colored into 2 colors: black and white. Of course, there are no roads between cities of the same color.

Recently, crime rate increased in the Kingdom. Gentlemen of the road take away money from poor wayfarers. To avoid this, the King decided to ask for help of one of the most famous communities in the Kingdom — the Research Institute of Guards in the High Towers (RIGHT). They can, with the help of the King and Kingdom's treasury, place guard towers in some cities. A guard tower placed in a city reduces crime rate on all roads incident to this city. Of course, crime rate on *all* roads need to be reduced — so the King has to select cities for guard towers in such a way that for every road there is a guard tower in at least one of the two cities that are connected by this road. Placing guard tower is *very* costly, so the King has to place the minimal possible number of towers.

Additionaly, placing tower in a city reduces crime rate in this city as well. The King doesn't need to reduce crime rate in all cities, but loyal citizens are pleased when crime rate in their city is reduced. The King wants to maximize the total pleasure — that is, the total population of cities with towers should be maximal possible.

So, to help the King and all the Kingdom, you have to write a program that selects an optimal subset of cities to place guard towers in them. The number of cities in this subset should be minimal possible. In case of ambiguity, the total population of these cities should be maximal possible. In case of ambiguity, output any such subset.

Input

The first line of the input file contains two integers — n and m ($1 \le n \le 300$, $1 \le m \le 10\,000$). Here, n is the number of cities in the Kingdom and m is the number of roads. The second line contains n positive integers — population of cities 1, 2, …, n, respectively. There is no city with population more than 100.

Next m lines contain description of roads. Line number $i + 2$ contains two integers — a_i and b_i. They mean that i-th road connect cities a_i and b_i. Cities are numbered from 1.

Output

The first line of the output file should contain two integers — P and K. Here, P is the total population of cities where towers should be placed and K is the number of such cities. Second and last line should contain K integers separated by spaces — the numbers of these cities in increasing order.

Example

standard input	standard output

4 4
1 2 3 4
1 2
1 4
2 3
3 4
	6 2
	2 4

Problem H. How to Pack String

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 Mebibytes

Consider a string s over an alphabet of N letters numbered 1, 2, …, N. For each letter i, Helge knows the frequency f_i which is how many times letter i occurs in string s. Helge should assign a non-empty code c_i consisting of zeroes and ones to each letter i. For different i and j, c_i could not be a prefix of c_j. Additionally, length of each code c_i could be no greater than a given integer L. Finally, after replacing each letter i of s by its code c_i, the length of the resulting encoded string should be the minimal possible provided that the restrictions above are met.

Help Helge to find the lengths of codes c_i.

Input

The input consists of one or more test cases.

The first line of each test case contains two positive integers N and L separated by a single space ($N \le 100$, $L \le 30$). The second line contains N integers f_i separated by single spaces ($1 \le f_i \le 1\,000\,000\,000$).

There are at most 500 test cases. The input will be terminated by a line containing two zeroes which should not be considered as a test case.

Output

For each test case, write two lines. On the first line, output a single integer which is the minimal possible length of the resulting string. On the second line, output N integers: the lengths of codes c_1, c_2, …, c_N. Separate consecutive integers by a single space.

It is guaranteed that at least one solution exists. In case there are multiple solutions, output any of them.

Example

standard input	standard output

```
1 1
1
8 3
1 1 2 3 5 8 13 21
7 7
1 2 3 5 8 13 21
0 0
```
	```
	1
	1
	162
	3 3 3 3 3 3 3 3
	124
	6 6 5 4 3 2 1
	```

Problem I. Integer Pairs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 Mebibytes

Given an array of integers $a_1, a_2, \ldots, a_n$ and a positive integer X, find the number of pairs of indices (i, j) such that $i < j$ and $\gcd(|a_i - a_j|, X) = 1$.

Here, $\gcd(p, q)$ stands for the greatest common divisor of p and q. Note that $\gcd(0, p) = p$ for any positive integer p.

Input

The input consists of one or more test cases.

Each test case starts with a line containing two integers: n, the number of elements of the array, and X ($1 \le n \le 10^5$, $1 \le X \le 10^8$). The next line contains n integers a_i ($1 \le a_i \le 10^9$).

The total number of elements in all arrays in the input will not exceed 10^5. There are at most 1000 test cases. The input will be terminated by a line containing two zeroes which should not be considered as a test case.

Output

For each test case, write a single line containing the number of pairs.

Example

standard input	standard output

7 30
1 2 3 4 5 6 7
0 0
	6

Problem J. Journey for Treasure

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

This is interactive problem.

Indiana Jones is on his next quest for the treasure. The treasure is buried on the field $2W \times 2H$. Indiana have a tool which tells the direction to the treasuse from current position. Unluckly, the tool is not so precise: Indiana gets an error of up to E degrees with that tool. Additionally, this tool uses batteries as source of power, so Indiana can do at most 200 measurements. Help him to find the treasure

Interaction Protocol

At the beginninng of the interaction process, the jury program prints three real numbers W, H and E ($0 \le W \le 10^4$, $0 \le H \le 10^4$, $0 \le E \le 120$. The numbers are given with 12 places after the decimal separator.

To use the tool, send the requests in the form "`? x y`", where x and y are coordinates of the position to use tool at ($-W \le x \le W$, $-H \le y \le H$). Dont forget to print end-of-line character and flush the output after the query.

The jury program gives you the direction in degrees with absolute error no more than E. The answer is generated in the range $[\alpha - E, \alpha + E]$ with uniform distribution (there α is direction to the treasure) and is does not exceed 180 degrees by absolute value

When you are ready to print the answer, output it in the format "`! X Y`". This is not counted as the query. The answer is considered correct it absolute error for each coordinate does not exceed 0.5.

Example

standard input	standard output
100 100 0	
	? -97.30147375 -55.03559390
45.8958081317	
	44.46472896 -54.50272726
110.928234444	
	! 4.50000018 50.00000005

Problem K. King of Renovations

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

Most of Berland's roads are not well-maintained.

The king of Berland, concerned by numerous requests of his subordinates, decided to renovate some of the roads with innovative technology. In Berland there are n cities numbered 1 through n. Some pairs of cities are connected with *unidirectional* roads. The chief builder of Berland has selected m roads which should be renovated and for each of those roads he estimated the renovation cost.

The king would like each citizen of Berland to feel the difference in the quality of the road system. He assumed that citizens of a city will be satisfied if it is possible to both enter the city and leave the city using a renovated road. Your task is to select which roads to renovate to minimize the total renovation cost, while satisfying all citizens of Berland.

Input

The first line of the input contains two integers n and m ($2 \le n \le 300$, $1 \le m \le n^2$) that represent the number of cities in Berland and the number of unidirectional roads which can be renovated. Each of the next m lines of the input contains three integers x, y and k ($1 \le x, y \le n$, $0 \le k \le 10^5$) describing a road from the city x to the city y, which renovation cost is k dollars. Each ordered pair x, y appears in the input at most once. Observe that there might be a road starting and ending in the same city.

Output

The output should consist of a single line containing the minimum renovation cost, subject to the constraints mentioned in the task description, or a single word "`NIE`" (Polish for *no*), if it is impossible to design a renovation plan fulfilling king's requirements.

Example

standard input	standard output

4 6
1 2 1
2 1 2
1 3 3
3 1 4
3 2 5
4 4 6
	16

4 4
1 2 5
2 3 4
3 1 8
2 4 7
	NIE

Problem L. Letters Arrangement

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Consider the next game for one player.

Given the string s, consisting of letters 'A', 'B' and exactly two sequential spaces.

At one step player is allowed to choose any two consequent non-space characters and swap those characters with pair of spaces, preserving the order.

The goal of the game is to transform the string s into the string, where all characters 'B' are going after all characters 'A', using no more than $|s|$ steps.

Input

First line of the input contains string s ($2 \le |s| \le 10^5$). Spaces are represented by underline characters ('_').

Output

IF it is impossible to solve the game, print -1. Otherwise in the first line print one integer k ($0 \le k \le |s|$) — number of steps. In the second line print the steps; i-th step is represented by position of the leftmost of the pair of spaces after this step. If there are more than one solution, print any of them.

Examples

standard input	standard output

`BB__AA`	`2`
	`1 5`

`AA__BB`	`0`

Problem M. Maze Generation

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A computer generates a 2-dimensional maze, consisting of walls and empty cells. The start is in the upper left cell, the finish is in the bottom right cell. Each cell has a fixed independent probability, that there will be a wall in it after the maze is generated. As soon as the generator puts randomly all the walls, it will check if there is a path from the upper left corner to the bottom right one, using only vertical and horizontal moves length of one cell. It is prohibited to go out from the maze. If such path exists, then the maze is considered correct and it is a result of the generator work. Else, the algorithm generates a maze again and again, until the correct maze is generated. Cells with the probability equal to 1.0 will never block off a path from the start to the finish, because in another way the correct maze will be never generated. The start and the finish can't contain a wall in a correct maze.

Your task: knowing the size of the maze and probabilities of setting walls in each cell, calculate the probability of locating a wall in each cell at the end of the algorithm work, i.e. the probability that there will be a wall in a correct maze.

Input

There are N and M "— height and width of the maze respectively, in the first line ($1 \le N, M \le 6$). In next N lines M real numbers in range [0.0; 1.0] with no more than one digit after decimal point are given. Each number is a probability of setting a wall in a corresponding cell at every iteration of the generator.

Output

Output N lines with M real numbers in each. Each number is a probability of setting a wall in a corresponding cell in the final correct maze accurate to 6 digits.

Example

standard input	standard output

2 2
0.5 1.0
0.5 0.5
	0.000000 1.000000
	0.000000 0.000000