Problem A. Airlines

Input file:	`airlines.in`
Output file:	`airlines.out`
Time limit:	2 seconds
Memory limit:	256 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`'вЪЭ™, 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

airlines.in	airlines.out
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:	`battlerobots.in`
Output file:	`battlerobots.out`
Time limit:	5 seconds
Memory limit:	256 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

battlerobots.in	battlerobots.out
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:	3 seconds
Memory limit:	256 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 query consists of an word "`Test`" and 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.

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 2 2 2 2 1 1 1	1.0 1.0 1.0 0.0 0.0 1.0 2.0 1.0 1.0 2.0 5.0 5.0 5.0 7.0 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:	`davyjones.in`
Output file:	`davyjones.out`
Time limit:	2 sec
Memory limit:	256 mebibytes

Captain Davy Jones (image on the right) 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

davyjones.in	davyjones.out
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. Ellipses

Input file:	`ellipses.in`
Output file:	`ellipses.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

For a non-degenerated triangle (triangle is non-degenerated, if each of its angles is strictly between 0 and 180 degrees) find the circumscribed ellipse with minimal area and inscribed ellipse with maximal area.

Input

Each of three lines of the input file contains two integers x and y — coordinates of some vertice of triangle. All coordinates does not exceed 10^5 by absolute value.

Output

In first line of the output file print parameters of the circumscribed ellipse — coordinates of the foci and sum of distances from foci to arbitrary point of ellipse. Foci can be listed in arbitrary order. In second line print parameters of the inscribed ellipse in the same format. Solution will be accepted, if absolute or relative error will be not greater than 10^{-4}. If there are multiple answers, print any of them.

Example

ellipses.in	ellipses.out
0 0 2 4 4 0	2.0000 0.0000 2.0000 2.6667 5.3333 2.0000 2.0000 2.0000 0.6667 2.6667

Problem F. Funny Graphs

Input file:	`funny.in`
Output file:	`funny.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

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

funny.in	funny.out
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. Generate the maze

Input file:	maze.in
Output file:	maze.out
Time limit:	2 seconds
Memory limit:	256 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

maze.in	maze.out
2 2 0.5 1.0 0.5 0.5	0.000000 1.000000 0.000000 0.000000

Problem H. Holidays

Input file:	`holidays.in`
Output file:	`holidays.out`
Time limit:	4 seconds
Memory limit:	256 mebibytes

Vasya is looking forward to his summer vacation, which he is going to spend lying on the golden sands of sea beaches. Taking into account his biorhythm, weather as well as cultural and soccer activities in his hometown, for each of the n days Vasya determined its recreation factor, which measures fun of a given day. Each of the factors is an integer, which might be negative — during such days Vasya would rather watch the soccer at home city.

Fortunately Vasya does not have to spend the whole vacation at the seaside. His favourite low-cost airlines prepared a special offer, which allows Vasya to buy k return tickets at a price of one ticket (each of those tickets can be used to travel from Vasya's home city to the seaside and back).

Your task is to help Vasya plan his holidays, that is to maximize the sum of the recreation factors of the days Vasya is going to spend at the seaside, assuming that during his vacation Vasya will fly to the seaside at most k times. For simplicity we assume that low-cost airlines have only overnight flights.

Input

The first line of the input contains two integers n and k ($1 \le k \le n \le 1\,000\,000$). The second line of the input contains n integers (with absolute values not greater than 10^9), describing the recreation factors of subsequent vacation days.

Output

The output should consist of a single line containing the sum of recreation factors in the optimum plan.

Example

holidays.in	holidays.out
5 2 7 -3 4 -9 5	13

Problem I. Interesting number

Input file:	`interesting.in`
Output file:	`interesting.out`
Time limit:	12 seconds
Memory limit:	256 mebibytes

— 13.69! What an interesting number! — exclaimed Bob on looking at the cash register receipt.

— Really? — asked Alice. — What's so interesting about 1369?

— 13 squared is 169, and both numbers can be found in the digits of 1369, — explained Bob.

— Big deal! — replied Alice. — 31 squared is 961, 19 squared is 361, 6 squared is 36, 3 squared is 9, 1 squared is 1. All of those numbers are in the digits of 1369 too.

— Wow! That makes 1369 even more interesting!

Write a program to help Bob find all of the hidden pairs of numbers and their squares in a given number.

Input

Input has a list of up to 60 non-negative integers, each less than 10^{13}, one per line. There will not be any leading zeros on the numbers. The end of input is indicated by the value 0. This value should not be processed.

Output

For each number, have a line with "`Hidden squares in `" and the original number. Then have the list of numbers and their squares with digits found in the original number, in ascending order. You should format your answer as shown in the sample output below and have a blank line after the output for each number.

A number is contained in the digits of a containing number if each digit in the number is a digit in the containing number. If a digit appears multiple times in the number, it must appear at least that many times in the containing number. So 100 is contained in 1000, but 100 is not contained in 10.

Example

interesting.in	interesting.out
1369 27 10 0	Hidden squares in 1369 1 * 1 = 1 3 * 3 = 9 6 * 6 = 36 13 * 13 = 169 19 * 19 = 361 31 * 31 = 961 Hidden squares in 27 Hidden squares in 10 0 * 0 = 0 1 * 1 = 1

Problem J. Journey of the King

Input file:	journey.in
Output file:	journey.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

A king wishes to go for a walk on a square chessboard with the following conditions:

1. Any two successive cells of the path must be adjacent, i.e., share an edge or a corner (thus, a cell may have up to eight adjacent cells).
2. Each cell must be visited exactly once; the first and the last cells of the path must coincide (thus, the first cell of the path will be actually visited twice).
3. The path must have no self intersections (if we think of a path as a closed polyline with vertices at cells' centers).

Your task is to find the maximal possible length of a king's path (here we mean the length of the polyline, not the number of king's moves).

Input

The only line of the input file contains an integer N ($1 \le N \le 300$), denoting the size of the chessboard.

Output

The only line of the output file must contain the length of the king's tour with at least three precise digits after the decimal point. The cells have side 1.

Examples

journey.in	journey.out
1	0.000
2	4.000
3	9.414

Problem K. King and Roads

Input file:	`kingandroads.in`
Output file:	`kingandroads.out`
Time limit:	7 seconds
Memory limit:	256 mebibytes

Most of Byteland's roads are not well-maintained. The king of Byteland, concerned by numerous requests of his subordinates, decided to renovate some of the roads. In Byteland there are n cities numbered 1 through n. Some pairs of cities are connected with *unidirectional* roads. The chief builder of Byteland 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 Byteland 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 Byteland.

Input

The first line of the input contains two integers n and m ($2 \leq n \leq 300$, $1 \leq m \leq n^2$) that represent the number of cities in Byteland 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 ($1 \leq x, y \leq n$, $0 \leq k \leq 10^5$) 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

kingandroads.in	kingandroads.out
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. Local Grocery

Input file:	`local.in`
Output file:	`local.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Your company, Calculation Solutions, which specializes in accounting software has been contacted by the local grocery for help. At the end of each day at the store, the owner, Mr. Smith has a list of transactions.

Each transaction is of the form:

1. p q: which means q items costing p dollars each were sold.
2. p: which means an item costing p dollars was sold.
3. $-p$: which means an item costing p dollars was returned.

Mr. Smith would like for you to help him by generating the revenue his store had on each day.

Input

The first line contains an integer N ($1 \leq N \leq 100$), the total number of days. Each day starts with an integer T ($0 \leq T \leq 100$) on a line, T being the total number of transactions on that day. It is followed by T lines containing transactions of type 1, 2 or 3. Since Mr. SmithвЂ™s store is mid-sized, p and q are relatively small ($1 \leq p, q \leq 10$). The transactions are in the format: "`1 p q`", "`2 p`" or "`3 -p`" (quotes for clarity).

Output

For each of N days, output the revenue R for that day in the format "`Day D: R dollars.`".

Example

local.in	local.out
3 2 1 4 5 2 6 1 3 -7 0	Day 1: 26 dollars. Day 2: -7 dollars. Day 3: 0 dollars.