Problem 2. Aaa

Input file:	`a.in`
Output file:	`a.out`
Time limit:	7 seconds
Memory limit:	256 Mebibytes

During one of hundreds of absolutely unimportant lectures, senior student was evenly distributing Sudoku puzzles around herself. Sasha was also given one, and that was a preposition to speak.

— I hope this is the most difficult level?

— Just for you.

— Well, well, — seeing a deep irony in her eyes and the fact that the first three lines are already filled.

— If you solve this one, I'll give you another.

— I won't just solve this, but I will say how many solutions exist, — said Sasha boastfully, not knowing what a problem he has created for himself.

— Well, well, we'll see.

The next two days Sasha was engaged only in looking for number of solutions for sudoku. By the way, sudoku is a puzzle in which it is proposed to fill a 9×9 table with numbers from 1 to 9 so that in every row, column and each of the 9 squares 3×3, all the numbers were different. Initially, some cells are already filled and you are to write the numbers in empty cells.

— And what was the girl's name, again?

— Oh...

Input

There are 3 lines, consisting of 9 integers from 1 to 9 in each. Those are first three lines of Sudoku.

Output

The only number: number of solutions. It is guaranteed that at least one solution exists.

Example

a.in	a.out

`1 4 7 5 6 3 8 2 9`
`2 5 8 4 7 9 1 3 6`
`3 6 9 1 2 8 5 4 7`
	`7013953152`

Problem 3. Bridges: The Final Battle

Input file:	`bridges3.in`
Output file:	`bridges3.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Does your research work have any practical applications?

Frequently asked question

Foreword

Serezha has almost finished his thesis. The subject hasn't changed since December: "Dynamic 2-Edge-Connectivity Problem". The algorithm has been invented and tested, the bound of $O(K \log K)$ has been proved. The only thing left is to write about "practical applications".

Well, does the problem "Dynamic 2-Edge-Connectivity Problem" indeed have any practical applications? That's not a simple question. Probably, it's even harder than the problem itself. Whatever, the thesis has to be done.

So, the first practical application: let us create a contest problem about it!

Problem

Given an undirected graph with no more than 10^5 vertices. Initially it does not contain any edges. You have to process requests `ADD x y` and `DEL x y` — to add and to remove edge from x to y, respectively.

After each request, you should find **the number of bridges** in the graph.

There are no multiedges and loops.

For every request to remove an edge, the corresponding edge exists.

Solution

The thesis has to be pretty hard to be written in five hours. So you are given five hints.

1. Requests to add or remove edge make the edge "alive" during some intervals of time.
2. Use "Divide and conquer" idea.
3. Compress components of biconnectivity.
4. Even having compressed biconnectivity components, the graph can be reduced provided there are few requests.
5. The solution in $O(K \log K)$ exists.

Input

The first line of input contains two integers N and K: the number of vertices and requests, respectively. $1 \leqslant N \leqslant 10^5$, $1 \leqslant K \leqslant 10^5$.

The following K lines contain requests, one per line. Each request starts with a word "`ADD`" or "`DEL`", depending on the type of the request. Two integers a_i and b_i follow, describing the edge to add or to remove. $1 \leqslant a_i, b_i \leqslant N$, $a_i \neq b_i$.

Output

Write K integers: the number of bridges in the graph after each request.

Example

bridges3.in	bridges3.out
4 8 ADD 1 2 ADD 2 3 ADD 1 3 DEL 2 3 DEL 1 2 ADD 2 4 ADD 1 4 ADD 2 3	1 2 0 2 1 2 3 0

Problem 5. Reflections

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds (4 seconds for Java)
Memory limit:	256 MB
Queries limit:	2 000

This is an interactive problem.

Consider N-dimensional linear space $\mathbb{R}^N$, where distance between points x and y is determined as follows. Let apply to both points linear mapping A, whose norm doesn't exceed 100, resulting in images $\tilde{x}$, $\tilde{y}$. So, distance between x and y equals to:

$$\sqrt{\sum_{i=1}^{N}(\tilde{x}_i - \tilde{y}_i)^2}.$$

You are drawing a polyline with a pen. Initially pen is placed at zero point (origin of the coordinate system). There are $2 \cdot N$ hyperplanes, each of which can be used as a symmetry axis to reflect the position of the pen. Also you can draw the next segment to know out its length.

The problem is to calculate the distance between given points a and b.

Interaction protocol

Initially you are given a number N, description of hyperplanes and coordinates of a and b points.

For each "reflect" command you'll get reflected position of the pen. For each "draw" command you'll get the length of drawn segment. For each answer output you'll get a judgement about correctness of your answer.

You should terminate your program after correct answer only. The absolute or relative error of 10^{-3} is allowed.

Output

Each line should contain the only command:

- `Reflect` x — reflect the pen over hyperplane x.
- `Request` — draw the segment of the polyline.
- `Answer` d — output the answer d — distance between a and b points.

The number of commands shouldn't exceed 2 000.

Don't forget to `flush` the standard output after printing each line.

Input

First line contains single integer N ($2 \leqslant N \leqslant 10$) — space dimension. It's folowed by description of $2 \cdot N$ hyperplanes: for each hyperplane there is line containing coordinates of hyperplane's normal followed by line containing coordinates of some point that belongs to hyperplane.

All numbers are integer. All normal's coordinates does not exceed 100 by it's absolute value, coordinates of points does not exceed 1 000 by it's absolute value. Normal of hyperplanes numbered from 1 up to N are linearly independent; the same is true for hyperplanes numbered from $N + 1$ up to $2 \cdot N$. No two hyperplanes are parallel. There is no hyperplain that contains origin of coordinate system.

Each of the following two lines contains N integers, that does not exceed 10 000 by it's absolute value — coordinates of points a and b respectively.

Each of the following lines contains answer to some request:

- `Reflect`: N real numbers with 9 signs after the decimal point — pen's coordinates.
- `Request`: single real number with 9 signs after the decimal point — segment's length.
- `Answer`: single symbol: «`N`» in case answer is incorrect, and «`Y`» otherwise.

Example

standard output	standard input
	2
	1 0
	2 0
	-1 1
	0 -1
	0 1
	0 -2
	1 1
	2 2
	4 2
	10 4
Answer 1	N
Answer 1.5	N
Reflect 1	4.000000000 0.000000000
Reflect 2	1.000000000 3.000000000
Reflect 1	3.000000000 3.000000000
Reflect 2	4.000000000 2.000000000
Request	8.944271910
Reflect 3	4.000000000 -6.000000000
Reflect 1	0.000000000 -6.000000000
Reflect 4	10.000000000 4.000000000
Request	12.649110641
Answer 12.64	Y

In the given sample mapping doubles the coordinate values.

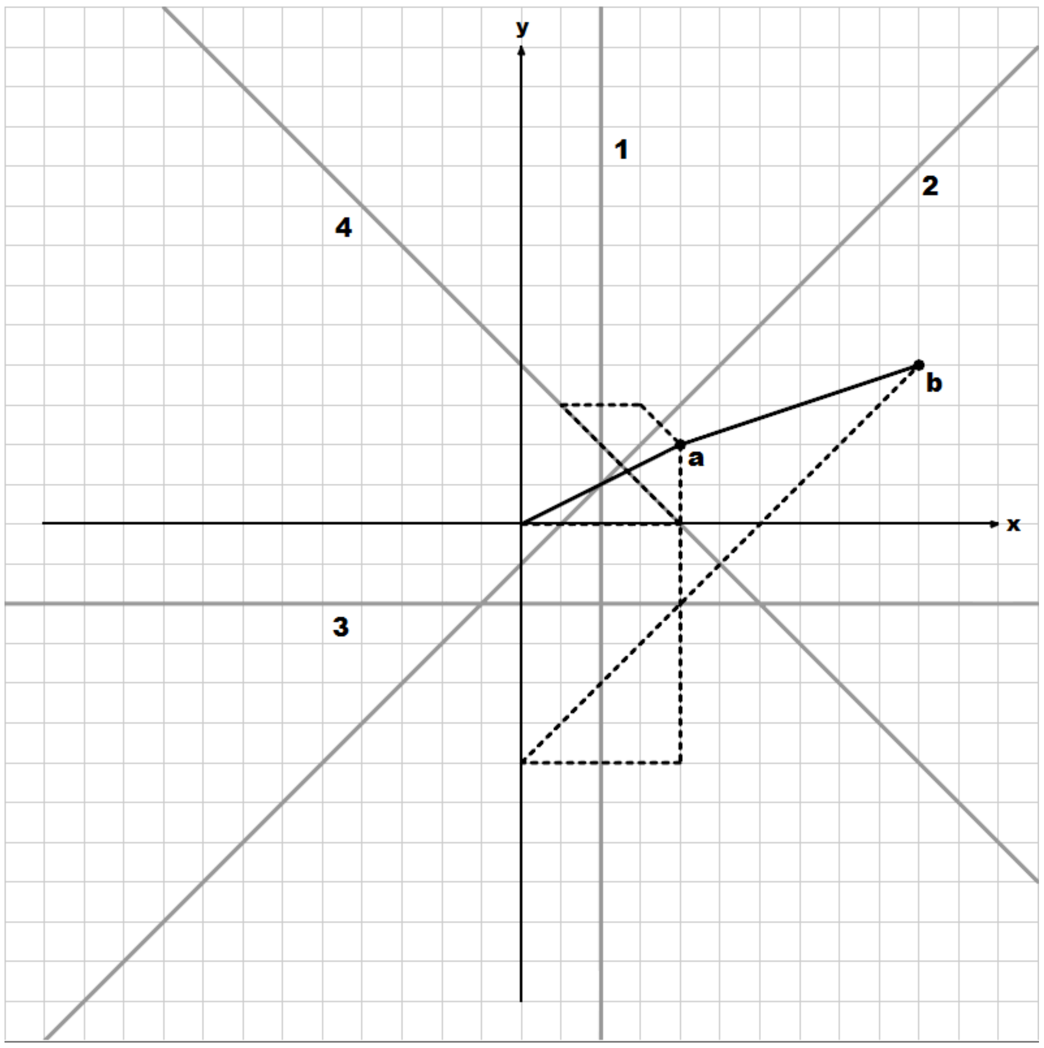
Note

Linear mapping $A : \mathbb{R}^N \to \mathbb{R}^N$ is mapping that satisfies to the linearity properties: $A(x+y) = A(x)+A(y)$ and $A(\alpha x) = \alpha A(x)$ for each $x, y \in \mathbb{R}^N$ and $\alpha \in \mathbb{R}$.

Norm of the linear mapping is supremum of value $||A(x)||$ for each point x that satisfies $||x|| = 1$. Here $||x||$ is the usual euclidian distance from origin to x.

*T*he set of vectors v_1, v_2, …, v_k is called Linearity independent if for any set α_1, α_2, …, α_k consisting of k numbers, at least one of which is not equal to zero, the vector $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_k v_k$ is not equal to zero vector.

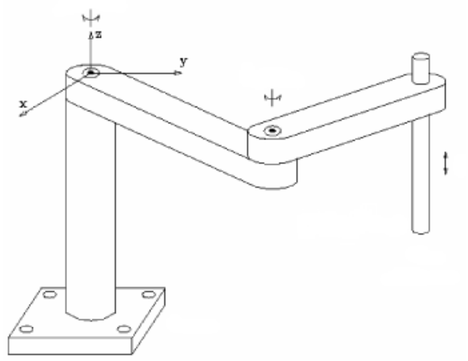
The following image corresponds to the sample:



Problem 7. Plotter

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

Pete's parents got him the Mindrack 2.0 kit for building robots. Pete read the manual and decided to build a simple plotter to draw shapes. The plotter is a mechanical arm consisting of two segments with a rotary joint and a slot for a pencil in the distal end. The arm is attached to a base using a joint too. Additionally the arm has servos that can rotate it relative to the base and turn segments relative to each other. The arm can draw various shapes on paper. Below is a schematic illustration:



Pete wrote a program that allows drawing various closed polylines based on a given set of points by means of coordinated actions of the servos. The arm can draw a polyline without taking the pencil off the paper. The only problem Pete has encountered is the limited mobility of the arm segments. The basal joint can turn no more than φ degrees left and right relative to its central position and the joint connecting the segments cannot turn more than ϑ degrees relative to the codirectional alignment of the segments. For that reason a situation can occur during drawing shapes when the robotic arm is unable to continue drawing without lifting the pencil from paper. In this case Pete has to disconnect the pencil manually, switch the arm into a new configuration and reconnect the pencil. The arm then continues drawing the polyline from the same point where it has ended drawing before manual switching. Naturally, Pete wants to minimize the amount of such manual switches.

Let the arm be positioned in the Oxy reference plane and attached to its origin. Then the central position of the arm is directed along the Oy axis. Assume the arm consists of two segments with the lengths L and l, corresponding to the basal segment attached to the base and distal segment holding the pencil. Given the parameters of the arm — the numbers L, l, φ and ϑ, and coordinates of polyline points on the plane you have to calculate the minimal number of manual arm switches necessary to draw the polyline or determine that it is impossible. The arm can begin drawing from any point of the polyline and in any of the two directions. However, once the start point and the drawing direction have been selected the latter cannot be changed. In case of the switch, the arm must continue drawing from the same point where switching has occurred and in the same direction as before. If the polyline has self-intersections or self-contacts, then the arm is not allowed to switch to another polyline part at such points, it still must draw polyline segments in the prescribed order taking into account the chosen direction and starting point.

Input

The first line of the input file contains four nonnegative numbers L, l, φ and ϑ ($1 \leqslant L + l \leqslant 10^4$, $\varphi + \vartheta \leqslant 180$). The next line contains an integer N — the number of points in the closed polyline ($3 \leqslant N \leqslant 10^5$).The next N lines contain pairs of integers x_i and y_i — Cartesian coordinates of polyline points. The absolute values of coordinates does not exceed 10^4. No two points coincide. It is assumed that the polyline consists of the segments

$[(x_1, y_1), (x_2, y_2)]$, $[(x_2, y_2), (x_3, y_3)], \ldots, [(x_{N-1}, y_{N-1}), (x_N, y_N)], [(x_N, y_N), (x_1, y_1)]$.

Output

The single line of the output file must contain the minimal number of manual switches of the arm necessary to draw the given polyline, or the number -1 if it is impossible to draw the polyline using the method described above.

Examples

plotter.in	plotter.out

4 4 90 90
6
-6 2
-3 6
3 6
6 2
3 7
-3 7
	1

1 1 45 45
3
1 1
5 1
1 5
	-1

Problem 11. 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 ($1 \leqslant k \leqslant n \leqslant 1\,000\,000$). 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 13. 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 ($3 \leqslant n \leqslant 80$). The second line contains two integer numbers w and h separated by whitespace: the dimensions of the room ($2 \leqslant w, h \leqslant 10^4$). 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 ($0 < x < w$, $0 < y < h$, $0 \leqslant |vx|, |vy| \leqslant 100$).

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	3.5
10 10	0.0
1 1 0 2	
8 8 -2 0	
8 6 0 1	
3	1.0
10 10	5.0
1 1 -1 3	
9 1 1 3	
5 3 0 2	
3	0.0
10 10	0.5
1 1 0 0	
1 2 0 0	
2 1 0 0	

Problem 17. Numbers

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

The most challenging exam for the students of the Recreational Mathematics Department is the exam in numeric shifts. Every year problems in this subject become more and more difficult. For the latest exam Professor Numberski has prepared something unbelievable.

The decimal notation of a certain number begins from a group of digits which will be denoted as B. In other words, the initial number is BX. If we multiply this number by A, the result of multiplication will be a number obtained by rewriting the group of digits B from the beginning to the end of the initial number, which means the final result of multiplication is XB.

The task of students is to find the minimal possible initial number BX based on given A and B.

Input

The only line contains two integers A and B ($2 \leqslant A \leqslant 9$, $1 \leqslant B \leqslant 10^6$).

Output

The only number — solution of the task, or `-1` if there is no solution.

Examples

numbers.in	numbers.out
`5 1`	`-1`
`3 1`	`142857`

Problem 19. 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 ($2 \leqslant h, w \leqslant 10$, $2 \leqslant k \leqslant 10$ and $1 \leqslant n \leqslant 1500$). 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 ($1 \leqslant r_i \leqslant h$, $1 \leqslant c_i \leqslant w$).

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 ($1 \leqslant s_i \leqslant k$). 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.