Beetle

A beetle finds itself on a thin horizontal branch. *"Here I am on a thin horizontal branch,"* thinks the beetle, *"I feel pretty much like on an x-axis!"* It surely is a beetle of deep mathematical thought.

There are also n drops of dew on that same branch, each holding m units of water. Their beetle–based integer coordinates are $x_1, x_2, \ldots, x_n$.

It is clear that the day will be hot. Already in one unit of time one unit of water goes away from each drop. The beetle is thirsty. It is so thirsty that if it reached a drop of dew it would drink it in zero time. In one unit of time the beetle can crawl one unit of length. But would all this crawling pay off? That's what buzzes the beetle.

So you are to write a program which, given coordinates of dew drops, calculates the *maximal* amount of water the beetle can possibly drink.

Input

The input is read from standard input. The first line contains two integers n and m. The next n lines contain integer coordinates $x_1, x_2, \ldots, x_n$.

Output

The program should write one line to standard output containing a single integer: the maximal amount of water the beetle can possibly drink.

Example

Input	Output
3 15 6 -3 1	25

Constraints

$0 \leq n \leq 300$, $1 \leq m \leq 1{,}000{,}000$, $-10{,}000 \leq x_1, x_2, \ldots, x_n \leq 10{,}000, x_i \neq x_j$ for $i \neq j$.

Candy Machine

In a candy factory, there is a mysterious machine. It produces delicious candies, each a little bit different from the others. The machine has a line of output slots, numbered 1 to n, from which the candies fall out as soon as they are ready. No one really knows how the machine works, but before it starts a production session, it prints a list telling the factory owner, when and from which slot each candy will fall out.

Now, the factory owner can install automatic wagons that move below the output slots to catch the falling candies. Of course, none of the candies should drop on the floor and get spoilt. However, since running the wagons is expensive, the owner would like to install as few wagons as possible.

Write a program that computes the minimum number of wagons needed to catch all candies. Moreover, your program shall output which wagon should catch which candy. The wagons run at a speed of one slot width per second. Before the production process starts, each wagon can be moved to the slot where it should catch its first candy.

Input

The input is read from standard input, and describes one production session of the machine. The first line contains exactly one integer n, the number of candies produced in that session. Each of the following n lines contains a pair of integers s_i and t_i, output slot and time for candy i. Each pair (s_i, t_i) is unique.

Output

Output should be written to standard output. The first line of the output contains exactly one integer w, the minimum number of wagons needed to catch all candies. The wagons are numbered from 1 to w. The following n lines indicate which wagon will catch which candy. For that purpose, each of these lines contains three integers: output slot s_j and output time t_j for some candy j and a wagon number $w(j)$, such that at time t_j wagon $w(j)$ will be at slot s_j and therefore be able to catch candy j.

Since all candies shall be caught, each slot and time pair given in one input line must appear in exactly one output line (in any order). If there is more than one solution, output any one.

Example

Input	Output
5 1 1 2 3 1 5 3 4 2 6	2 1 1 1 2 3 1 1 5 2 3 4 1 2 6 2

Constraints

$1 \leq n \leq 100\,000$
$0 \leq s_i, t_i \leq 1\,000\,000\,000$

Grading

For test cases worth 20% of the total score: $n \leq 85$ and $w \leq 4$.
For test cases worth 60% of the total score: $n \leq 8\,000$.

Subway Signalling Error

The subway in Stockholm consists of several subway lines. In this problem we will consider one line in particular, and the problems that quite often happens on it when there's been a "signalling error".

A subway line can be seen as two parallel rails connected at the endpoints. In the upper rail the trains are going from right to left and in the lower rail the trains are going from left to right. When a train reaches an endpoint of a rail, it switches to the opposite rail and changes direction. This switch is instantaneous and takes no time.

During normal traffic, the traffic flow is continuous and the trains are moving at a constant speed (1 length unit per time unit). The trains are evenly distributed; that is, at any given position on a rail, the trains will appear periodically. We will assume that the time it takes for a train to stop and pick up passengers is negligible.

Now, because of signalling errors, the trains have been randomly distributed along the line. Your job as the traffic manager is to, as fast as possible, ensure that the trains will be evenly distributed along the line again. Write a program that, given the current train positions, calculates how fast this can be achieved. You are allowed to order the trains to temporarily stop, and/or change direction anywhere along the line. If a train changes direction, it will move from one rail to the other.

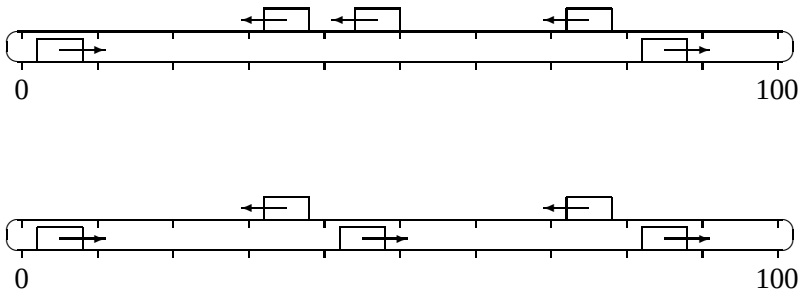

Figure 1: *Both rails have the length 100. In the upper example the trains are positioned at 5 (direction right), 35 (left), 46 (left), 75 (left) and 85 (right). One way to make the trains evenly distributed is for the train at position 46 to travel one unit to the left and then change direction. This takes one time unit. However, this is not the optimal solution; see Example 1 below.*

Input

The input is read from standard input. The first line in the input contains two integers separated by a single space, the length m of the subway rails, and the number n of trains on the line. Then follows n lines each describing the current position of a train. This will be given as an integer x_i and a direction (`L` for left, `R` for right), separated by a single space.

Output

Your program should output a single line to standard output, containing the shortest time it takes to make the trains evenly distributed. Your program should have an absolute precision error of at most

10^{-6}.

Example 1

Input	Output
100 5 5 R 35 L 46 L 75 L 85 R	0.5

Example 2

Input	Output
100 8 9 L 15 R 41 L 33 L 81 R 33 R 100 L 97 R	15.500000

Constraints

$100 \leq m \leq 100{,}000{,}000$

$1 \leq n \leq 100{,}000$

$0 \leq x_i \leq m$

Grading

For test cases worth 50% of the total score, $n \leq 200$