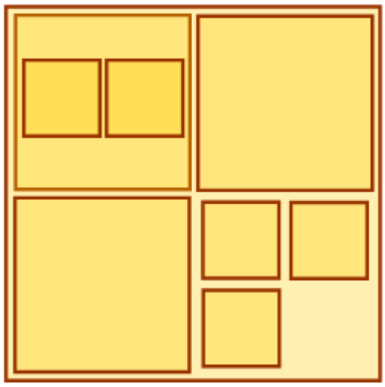
Codeforces Round #165 (Div. 1)

A. Magical Boxes

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Emuskald is a well-known illusionist. One of his trademark tricks involves a set of magical boxes. The essence of the trick is in packing the boxes inside other boxes.

From the top view each magical box looks like a square with side length equal to 2^k (k is an integer, $k \geq 0$) units. A magical box v can be put inside a magical box u, if side length of v is strictly less than the side length of u. In particular, Emuskald can put 4 boxes of side length 2^{k-1} into one box of side length 2^k, or as in the following figure:



Emuskald is about to go on tour performing around the world, and needs to pack his magical boxes for the trip. He has decided that the best way to pack them would be inside another magical box, but magical boxes are quite expensive to make. Help him find the smallest magical box that can fit all his boxes.

Input

The first line of input contains an integer n ($1 \leq n \leq 10^5$), the number of different sizes of boxes Emuskald has. Each of following n lines contains two integers k_i and a_i ($0 \leq k_i \leq 10^9$, $1 \leq a_i \leq 10^9$), which means that Emuskald has a_i boxes with side length 2^{k_i}. It is guaranteed that all of k_i are distinct.

Output

Output a single integer p, such that the smallest magical box that can contain all of Emuskald's boxes has side length 2^p.

Sample test(s)

input
2 0 3 1 5
output
3

input
1 0 4
output
1

input
2 1 10 2 2
output
3

Note

Picture explanation. If we have 3 boxes with side length 2 and 5 boxes with side length 1, then we can put all these boxes inside a box with side length 4, for example, as shown in the picture.

In the second test case, we can put all four small boxes into a box with side length 2.

B. Greenhouse Effect

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Emuskald is an avid horticulturist and owns the world's longest greenhouse — it is effectively infinite in length.

Over the years Emuskald has cultivated n plants in his greenhouse, of m different plant species numbered from 1 to m. His greenhouse is very narrow and can be viewed as an infinite line, with each plant occupying a single point on that line.

Emuskald has discovered that each species thrives at a different temperature, so he wants to arrange $m - 1$ borders that would divide the greenhouse into m sections numbered from 1 to m from left to right with each section housing a single species. He is free to place the borders, but in the end all of the i-th species plants must reside in i-th section from the left.

Of course, it is not always possible to place the borders in such way, so Emuskald needs to replant some of his plants. He can remove each plant from its position and place it anywhere in the greenhouse (at **any** real coordinate) with no plant already in it. Since replanting is a lot of stress for the plants, help Emuskald find the minimum number of plants he has to replant to be able to place the borders.

Input

The first line of input contains two space-separated integers n and m ($1 \le n, m \le 5000, n \ge m$), the number of plants and the number of different species. Each of the following n lines contain two space-separated numbers: one integer number s_i ($1 \le s_i \le m$), and one real number x_i ($0 \le x_i \le 10^9$), the species and position of the i-th plant. Each x_i will contain no more than 6 digits after the decimal point.

It is guaranteed that all x_i are different; there is at least one plant of each species; the plants are given in order "from left to the right", that is in the ascending order of their x_i coordinates ($x_i < x_{i+1}, 1 \le i < n$).

Output

Output a single integer — the minimum number of plants to be replanted.

Sample test(s)

input
3 2 2 1 1 2.0 1 3.100
output
1

input
3 3 1 5.0 2 5.5 3 6.0
output
0

input
6 3 1 14.284235 2 17.921382 1 20.328172 3 20.842331 1 25.790145 1 27.204125
output
2

Note

In the first test case, Emuskald can replant the first plant to the right of the last plant, so the answer is 1.

In the second test case, the species are already in the correct order, so no replanting is needed.

C. Flawed Flow

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Emuskald considers himself a master of flow algorithms. Now he has completed his most ingenious program yet — it calculates the maximum flow in an undirected graph. The graph consists of n vertices and m edges. Vertices are numbered from 1 to n. Vertices 1 and n being the source and the sink respectively.

However, his max-flow algorithm seems to have a little flaw — it only finds the flow volume for each edge, but not its direction. Help him find for each edge the direction of the flow through this edges. Note, that the resulting flow should be correct maximum flow.

More formally. You are given an undirected graph. For each it's undirected edge (a_i, b_i) you are given the flow volume c_i. You should direct all edges in such way that the following conditions hold:

1. for each vertex v $(1 < v < n)$, sum of c_i of incoming edges is equal to the sum of c_i of outcoming edges;
2. vertex with number 1 has no incoming edges;
3. the obtained directed graph **does not have cycles**.

Input

The first line of input contains two space-separated integers n and m ($2 \le n \le 2 \cdot 10^5$, n - $1 \le m \le 2 \cdot 10^5$), the number of vertices and edges in the graph. The following m lines contain three space-separated integers a_i, b_i and c_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le c_i \le 10^4$), which means that there is an undirected edge from a_i to b_i with flow volume c_i.

It is guaranteed that there are no two edges connecting the same vertices; the given graph is connected; a solution always exists.

Output

Output m lines, each containing one integer d_i, which should be 0 if the direction of the i-th edge is $a_i \rightarrow b_i$ (the flow goes from vertex a_i to vertex b_i) and should be 1 otherwise. The edges are numbered from 1 to m in the order they are given in the input.

If there are several solutions you can print any of them.

Sample test(s)

input

```
3 3
3 2 10
1 2 10
3 1 5
```

output

```
1
0
1
```

input

```
4 5
1 2 10
1 3 10
2 3 5
4 2 15
3 4 5
```

output

```
0
0
1
1
0
```

Note

In the first test case, 10 flow units pass through path $1 \rightarrow 2 \rightarrow 3$, and 5 flow units pass directly from source to sink: $1 \rightarrow 3$.

D. Maximum Waterfall

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Emuskald was hired to design an artificial waterfall according to the latest trends in landscape architecture. A modern artificial waterfall consists of multiple horizontal panels affixed to a wide flat wall. The water flows down the top of the wall from panel to panel until it reaches the bottom of the wall.

The wall has height t and has n panels on the wall. Each panel is a horizontal segment at height h_i which begins at l_i and ends at r_i. The i-th panel connects the points (l_i, h_i) and (r_i, h_i) of the plane. The top of the wall can be considered a panel connecting the points $(-10^9, t)$ and $(10^9, t)$. Similarly, the bottom of the wall can be considered a panel connecting the points $(-10^9, 0)$ and $(10^9, 0)$. No two panels share a common point.

Emuskald knows that for the waterfall to be aesthetically pleasing, it can flow from panel i to panel j ($i \to j$) only if the following conditions hold:

1. $max(l_i, l_j) < min(r_i, r_j)$ (horizontal projections of the panels overlap);
2. $h_j < h_i$ (panel j is below panel i);
3. there is no such panel k ($h_j < h_k < h_i$) that the first two conditions hold for the pairs (i, k) and (k, j).

Then the **flow** for $i \to j$ is equal to $min(r_i, r_j) - max(l_i, l_j)$, the length of their horizontal projection overlap.

Emuskald has decided that in his waterfall the water will flow in a single path from top to bottom. If water flows to a panel (except the bottom of the wall), the water will fall further to **exactly one** lower panel. The total amount of water flow in the waterfall is then defined as the minimum horizontal projection overlap between two consecutive panels in the path of the waterfall. Formally:

1. the waterfall consists of a single path of panels $top \to p_1 \to p_2 \to \ldots \to bottom$;
2. the flow of the waterfall is the minimum flow in the path $top \to p_1 \to p_2 \to \ldots \to bottom$.

To make a truly great waterfall Emuskald must maximize this water flow, but there are too many panels and he is having a hard time planning his creation. Below is an example of a waterfall Emuskald wants:

Help Emuskald maintain his reputation and find the value of the maximum possible water flow.

Input

The first line of input contains two space-separated integers n and t ($1 \le n \le 10^5$, $2 \le t \le 10^9$), the number of the panels excluding the top and the bottom panels, and the height of the wall. Each of the n following lines contain three space-separated integers h_i, l_i and r_i ($0 < h_i < t$, $-10^9 \le l_i < r_i \le 10^9$), the height, left and right ends of the i-th panel segment.

It is guaranteed that no two segments share a common point.

Output

Output a single integer — the maximum possible amount of water flow in the desired waterfall.

Sample test(s)

input

```
5 6
4 1 6
3 2 7
5 9 11
3 10 15
1 13 16
```

output

```
4
```

input

```
6 5
4 2 8
3 1 2
2 2 3
2 6 12
1 0 7
1 8 11
```

output
2

Note

The first test case corresponds to the picture.

E. String Theory

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Emuskald is an innovative musician and always tries to push the boundaries of music production. Now he has come up with an idea for a revolutionary musical instrument — a rectangular harp.

A rectangular harp is a rectangle $n \times m$ consisting of n rows and m columns. The rows are numbered 1 to n from top to bottom. Similarly the columns are numbered 1 to m from left to right. String pins are spaced evenly across every side, one per unit. Thus there are n pins on the left and right sides of the harp and m pins on its top and bottom. The harp has exactly $n + m$ different strings, each string connecting two different pins, each on a different side of the harp.

Emuskald has ordered his apprentice to construct the first ever rectangular harp. However, he didn't mention that no two strings can cross, otherwise it would be impossible to play the harp. Two strings cross if the segments connecting their pins intersect. To fix the harp, Emuskald can perform operations of two types:

1. pick two different columns and swap their pins on each side of the harp, not changing the pins that connect each string;
2. pick two different rows and swap their pins on each side of the harp, not changing the pins that connect each string;

In the following example, he can fix the harp by swapping two columns:

Help Emuskald complete his creation and find the permutations how the rows and columns of the harp need to be rearranged, or tell that it is impossible to do so. He can detach and reattach each string to its pins, so the physical layout of the strings doesn't matter.

Input

The first line of input contains two space-separated integers numbers n and m ($1 \le n, m \le 10^5$), the height and width of the harp in units. Each of the following $n + m$ lines contains 4 space-separated tokens, describing a single string: two symbols a_i, b_i and two integer numbers p_i, q_i. The pair a_i, p_i describes the first pin, and the pair b_i, q_i describes the second pin of the string;

A pair s, x describes the position of a single pin in a following way:

1. s is equal to one of the symbols "L", "T", "R" or "B" (without quotes), which means that the pin is positioned on the left, top, right or bottom side of the harp accordingly;
2. x is equal to the number of the row, if the pin is on the left or right border of the harp, and to the number of the column, if the pin is on the top or bottom border of the harp.

It is guaranteed that no two different strings are connected to the same pin.

Output

If it is possible to rearrange the rows and columns to fix the harp, on the first line output n space-separated integers — the old numbers of rows now placed from top to bottom in the fixed harp. On the second line, output m space-separated integers — the old numbers of columns now placed from left to right in the fixed harp.

If it is impossible to rearrange the rows and columns to fix the harp, output "No solution" (without quotes).

Sample test(s)

input
3 4 L T 1 3 L B 2 2 L B 3 3 T R 1 2 T B 2 1 T R 4 1 B R 4 3
output
1 2 3 3 2 1 4

input
3 3 L T 1 1 T R 3 1

```
R B 3 3
B L 1 3
L R 2 2
T B 2 2
```

output

```
No solution
```