A. Okabe and Future Gadget Laboratory

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

Okabe needs to renovate the *Future Gadget Laboratory* after he tried doing some crazy experiments! The lab is represented as an n by n square grid of integers. A *good* lab is defined as a lab in which every number not equal to 1 can be expressed as the sum of a number in the same row and a number in the same column. In other words, for every x, y such that $1 \le x, y \le n$ and $a_{x,y} \ne 1$, there should exist two indices s and t so that $a_{x,y} = a_{x,s} + a_{t,y}$, where $a_{i,j}$ denotes the integer in i-th row and j-th column.

Help Okabe determine whether a given lab is *good*!

Input

The first line of input contains the integer n $(1 \le n \le 50)$ — the size of the lab.

The next n lines contain n space-separated integers denoting a row of the grid. The j-th integer in the i-th row is $a_{i,j}$ $(1 \le a_{i,j} \le 10^5)$.

Output

Print "Yes" if the given lab is *good* and "No" otherwise.

You can output each letter in upper or lower case.

Examples

input

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

output

```
Yes
```

input

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

output

```
No
```

Note

In the first sample test, the 6 in the bottom left corner is valid because it is the sum of the 2 above it and the 4 on the right. The same holds for every number not equal to 1 in this table, so the answer is "Yes".

In the second sample test, the 5 cannot be formed as the sum of an integer in the same row and an integer in the same column. Thus the answer is "No".

B. Okabe and Banana Trees

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

Okabe needs bananas for one of his experiments for some strange reason. So he decides to go to the forest and cut banana trees.

Consider the point (x, y) in the 2D plane such that x and y are integers and $0 \le x, y$. There is a tree in such a point, and it has $x + y$ bananas. There are no trees nor bananas in other points. Now, Okabe draws a line with equation $y = -\frac{x}{m} + b$. Okabe can select a single rectangle with axis aligned sides with all points on or under the line and cut all the trees in all points that are inside or on the border of this rectangle and take their bananas. Okabe's rectangle can be degenerate; that is, it can be a line segment or even a point.

Help Okabe and find the maximum number of bananas he can get if he chooses the rectangle wisely.

Okabe is sure that the answer does not exceed 10^{18}. You can trust him.

Input

The first line of input contains two space-separated integers m and b ($1 \le m \le 1000$, $1 \le b \le 10000$).

Output

Print the maximum number of bananas Okabe can get from the trees he cuts.

Examples

input

```
1 5
```

output

```
30
```

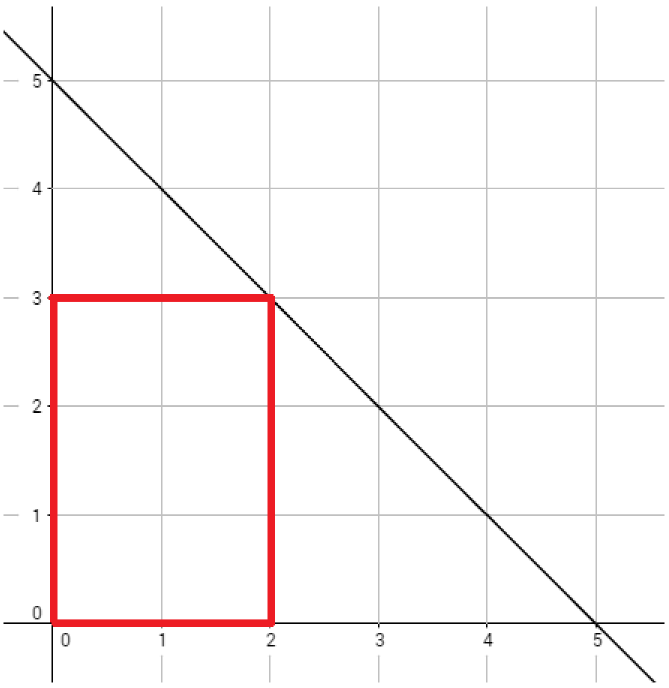
input

```
2 3
```

output

```
25
```

Note



The graph above corresponds to sample test 1. The optimal rectangle is shown in red and has 30 bananas.

C. Okabe and Boxes

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Okabe and Super Hacker Daru are stacking and removing boxes. There are n boxes numbered from 1 to n. Initially there are no boxes on the stack.

Okabe, being a control freak, gives Daru $2n$ commands: n of which are to add a box to the top of the stack, and n of which are to remove a box from the top of the stack and throw it in the trash. Okabe wants Daru to throw away the boxes in the order from 1 to n. Of course, this means that it might be impossible for Daru to perform some of Okabe's *remove* commands, because the required box is not on the top of the stack.

That's why Daru can decide to wait until Okabe looks away and then reorder the boxes in the stack in any way he wants. He can do it at any point of time between Okabe's commands, but he can't add or remove boxes while he does it.

Tell Daru the minimum number of times he needs to reorder the boxes so that he can successfully complete all of Okabe's commands. It is guaranteed that every box is added before it is required to be removed.

Input

The first line of input contains the integer n ($1 \le n \le 3 \cdot 10^5$) — the number of boxes.

Each of the next $2n$ lines of input starts with a string "`add`" or "`remove`". If the line starts with the "`add`", an integer x ($1 \le x \le n$) follows, indicating that Daru should add the box with number x to the top of the stack.

It is guaranteed that exactly n lines contain "`add`" operations, all the boxes added are distinct, and n lines contain "`remove`" operations. It is also guaranteed that a box is always added before it is required to be removed.

Output

Print the minimum number of times Daru needs to reorder the boxes to successfully complete all of Okabe's commands.

Examples

input

```
3
add 1
remove
add 2
add 3
remove
remove
```

output

```
1
```

input

```
7
add 3
add 2
add 1
remove
add 4
remove
remove
remove
add 6
add 7
add 5
remove
remove
remove
```

output

```
2
```

Note

In the first sample, Daru should reorder the boxes after adding box 3 to the stack.

In the second sample, Daru should reorder the boxes after adding box 4 and box 7 to the stack.

D. Okabe and City

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Okabe likes to be able to walk through his city on a path lit by street lamps. That way, he doesn't get beaten up by schoolchildren.

Okabe's city is represented by a 2D grid of cells. Rows are numbered from 1 to n from top to bottom, and columns are numbered 1 to m from left to right. Exactly k cells in the city are lit by a street lamp. It's guaranteed that the top-left cell is lit.

Okabe starts his walk from the top-left cell, and wants to reach the bottom-right cell. Of course, Okabe will only walk on lit cells, and he can only move to adjacent cells in the up, down, left, and right directions. However, Okabe can also temporarily light all the cells in any single row or column at a time if he pays 1 coin, allowing him to walk through some cells not lit initially.

Note that Okabe can only light a single row or column at a time, and has to pay a coin every time he lights a new row or column. To change the row or column that is temporarily lit, he must stand at a cell that is lit initially. Also, once he removes his temporary light from a row or column, all cells in that row/column not initially lit are now not lit.

Help Okabe find the minimum number of coins he needs to pay to complete his walk!

Input

The first line of input contains three space-separated integers n, m, and k ($2 \le n, m, k \le 10^4$).

Each of the next k lines contains two space-separated integers r_i and c_i ($1 \le r_i \le n$, $1 \le c_i \le m$) — the row and the column of the i-th lit cell.

It is guaranteed that all k lit cells are distinct. It is guaranteed that the top-left cell is lit.

Output

Print the minimum number of coins Okabe needs to pay to complete his walk, or `-1` if it's not possible.

Examples

input

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

output

```
2
```

input

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

output

```
-1
```

input

```
2 2 4
1 1
1 2
2 1
2 2
```

output

```
0
```

input

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

output

```
3
```

Note

In the first sample test, Okabe can take the path $(1, 1) \to (2, 1) \to (2, 3) \to (3, 3) \to (4, 3) \to (4, 4)$, paying only when moving to $(2, 3)$ and $(4, 4)$.

In the fourth sample, Okabe can take the path $(1,1) \to (1,2) \to (2,2) \to (3,2) \to (3,3) \to (3,4) \to (4,4) \to (5,4) \to (5,5)$, paying when moving to $(1, 2)$, $(3, 4)$, and $(5, 4)$.

E. Okabe and El Psy Kongroo

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

Okabe likes to take walks but knows that spies from the Organization could be anywhere; that's why he wants to know how many different walks he can take in his city safely. Okabe's city can be represented as all points (x, y) such that x and y are non-negative. Okabe starts at the origin (point $(0, 0)$), and needs to reach the point $(k, 0)$. If Okabe is currently at the point (x, y), in one step he can go to $(x + 1, y + 1)$, $(x + 1, y)$, or $(x + 1, y - 1)$.

Additionally, there are n horizontal line segments, the i-th of which goes from $x = a_i$ to $x = b_i$ inclusive, and is at $y = c_i$. It is guaranteed that $a_1 = 0$, $a_n \le k \le b_n$, and $a_i = b_{i - 1}$ for $2 \le i \le n$. The i-th line segment forces Okabe to walk with y-value in the range $0 \le y \le c_i$ when his x value satisfies $a_i \le x \le b_i$, or else he might be spied on. This also means he is required to be under two line segments when one segment ends and another begins.

Okabe now wants to know how many walks there are from the origin to the point $(k, 0)$ satisfying these conditions, modulo $10^9 + 7$.

Input

The first line of input contains the integers n and k ($1 \le n \le 100$, $1 \le k \le 10^{18}$) — the number of segments and the destination x coordinate.

The next n lines contain three space-separated integers a_i, b_i, and c_i ($0 \le a_i < b_i \le 10^{18}$, $0 \le c_i \le 15$) — the left and right ends of a segment, and its y coordinate.

It is guaranteed that $a_1 = 0$, $a_n \le k \le b_n$, and $a_i = b_{i - 1}$ for $2 \le i \le n$.

Output

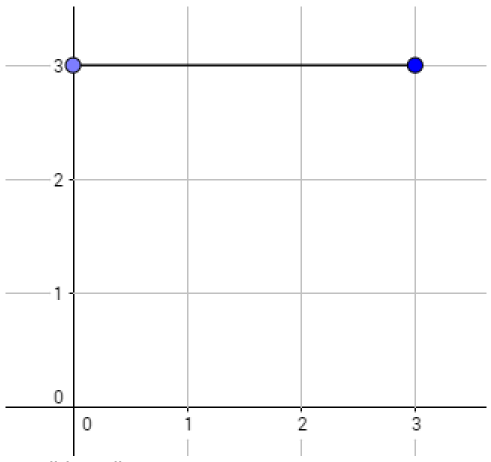
Print the number of walks satisfying the conditions, modulo 1000000007 ($10^9 + 7$).

Examples

input
```
1 3
0 3 3
```
output
```
4
```

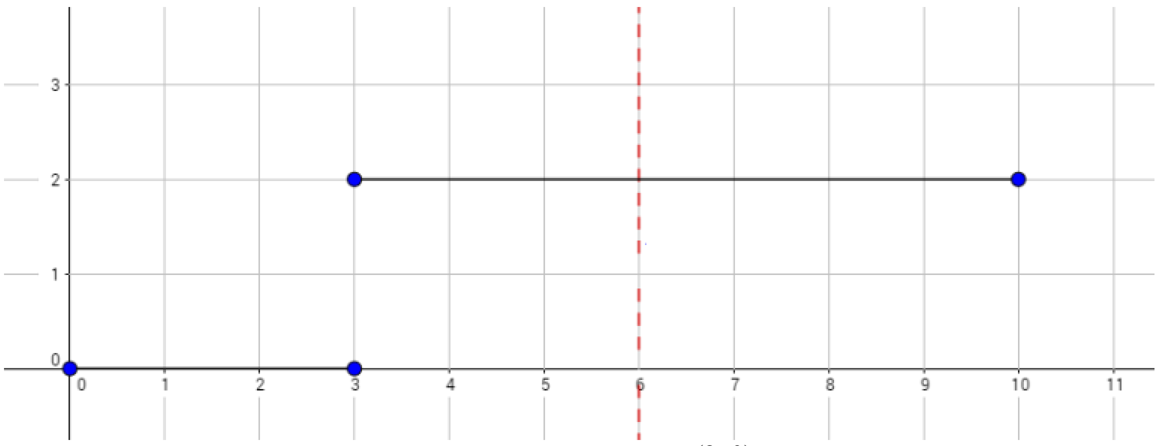
input
```
2 6
0 3 0
3 10 2
```
output
```
4
```

Note



The graph above corresponds to sample 1. The possible walks are:

- $(0, 0) \to (1, 0) \to (2, 0) \to (3, 0)$
- $(0, 0) \to (1, 1) \to (2, 0) \to (3, 0)$
- $(0, 0) \to (1, 0) \to (2, 1) \to (3, 0)$
- $(0, 0) \to (1, 1) \to (2, 1) \to (3, 0)$



The graph above corresponds to sample 2. There is only one walk for Okabe to reach $(3, 0)$. After this, the possible walks are:

- $(3, 0) \to (4, 0) \to (5, 0) \to (6, 0)$
- $(3, 0) \to (4, 0) \to (5, 1) \to (6, 0)$
- $(3, 0) \to (4, 1) \to (5, 0) \to (6, 0)$
- $(3, 0) \to (4, 1) \to (5, 1) \to (6, 0)$