Problem A. Star triangles

Input file:	`triangles.in`
Output file:	`triangles.out`
Time limit:	2 seconds
Memory limit:	256 megabytes
Detailed Feedback:	full

Zhomart likes watching the stars and construct a variety of geometric shapes from them. The sky is represented in the form of the Cartesian coordinate system, and the stars are represented by points on it. At this moment Zhomart interested in the question: how many different right triangles whose legs are parallel to the axes, you can create with the help of stars in the sky.

Input

In the first line of the input line you are given N — the number of stars on the sky ($3 \le N \le 300000$). Each of the next N lines contains integer X and Y ($|X, Y| \le 10^9$) — coordinates of the appropriate star.

Output

Print one number – the answer to the question.

Examples

triangles.in	triangles.out
3 0 0 1 0 0 1	1
4 0 0 1 0 0 1 1 1	4

For this problem you will have full feedback.

30% of tests contain $N \le 100$.

Problem B. Hyper-minimum

Input file:	`hyper.in`
Output file:	`hyper.out`
Time limit:	2 seconds
Memory limit:	256 megabytes
Detailed Feedback:	none

There is a 4-dimensional array X, each index of which is in interval from 1 to N. Your task is to construct new 4-dimensional array Y, elements of which can be calculated using the next formula: $Y[i_1, i_2, i_3, i_4] = min(X[j_1, j_2, j_3, j_4])$, where $1 \le i_k \le N - M + 1, i_k \le j_k \le i_k + M - 1$, and M is given.

Input

First line of the input file contains N and M ($1 \le M \le N$). Next lines of the input file contain elements of array X. The number of elements will be not more than 1500000 and elements will be integers not exceeding 10^9 by absolute value. They are given in such order, that the array can be read using following pseudocode:

```
for i = 1 to N:
    for j = 1 to N:
        for k = 1 to N:
            for l = 1 to N:
                read X[i, j, k, l]
```

Output

Output array Y in the same format as the X was given.

Examples

hyper.in	hyper.out
1 1 1	1
3 2 3 1 4 -4 0 4 0 0 -3 0 -2 -5 5 3 5 -4 4 -3 -5 -4 -4 5 -1 0 -3 -2 -1 2 -5 -5 -1 1 1 -4 3 5 3 -3 -3 3 0 1 4 -1 -2 3 -2 5 4 -1 -5 3 -4 0 -3 -1 3 -1 4 4 -1 -5 -3 4 -4 5 1 5 -4 3 2 2 -2 -2 4 2 -4 -3 1 3 1	-5 -5 -4 -3 -5 -5 -4 -5 -5 -5 -5 -5 -4 -5 -4 -5

Problem C. Energetic turtle

Input file:	`turtle.in`
Output file:	`turtle.out`
Time limit:	2 seconds
Memory limit:	256 megabytes
Detailed Feedback:	none

There is a grid with $N + 1$ rows and $M + 1$ columns. The turtle, which is on the cell (0, 0), wants to get into the cell (N, M). The turtle can only go up or right. There are K traps on the grid. If the turtle will get to one of these traps, it will turn up. The turtle has strength to stand up no more than T times. Calculate, how many different ways the turtle can reach the cell (N, M). Since this number can be very large, output the remainder of his division by Z.

Input

The first line contains 5 integers N, M, K, T and Z ($1 \le N, M \le 300000$, $0 \le K, T \le 20$, $1 \le Z \le 1000000000$). Each of the following K lines contains coordinates of a cell with a trap: X, Y ($0 \le X \le N$, $0 \le Y \le M$). It's guaranteed that all traps situated in different cells and there is no trap in cells (0, 0) and (N, M).

Output

Print one number – the answer.

Examples

turtle.in	turtle.out
1 1 1 0 1000 0 1	1
2 2 0 0 10	6

40% of tests contain $N, M \le 1000$

Problem D. Weighting stones

Input file:	`stones.in`
Output file:	`stones.out`
Time limit:	1 second
Memory limit:	256 megabytes
Detailed Feedback:	none

Jack somehow found N stones and arranged them in increasing order of their weights. No two weights are equal. The lightest stone is given the rank 1, the next lightest — 2, and so on, the heaviest stone gets the rank N.

He has a balance scale and decided to put all the stones on it's sides in some order. It's known in which order he is going to put those stones on the scale and on which side each stone gets.

You have to determine the state of scale after each stone is added. Jack doesn't tell the exact weights of those stones.

Input

The first line contains integer number N ($1 \le N \le 100000$).

Each of the next N lines contains two integer numbers: R ($1 \le R \le N$) and S ($1 \le S \le 2$). R is the rank of the next stone which is put on side S. All R's will be distinct.

Output

Output N lines — one for each added stone. If after adding the corresponding stone side 1 is heavier, output "<". If side 2 is heavier, output ">". If it's not clear in which state the scale will be, output "?".

Examples

stones.in	stones.out
5 1 2 3 1 2 1 4 2 5 1	< > > ? >