Just-in-Time Render Analysis

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Websites contain various elements, including visible UI elements and invisible grouping frames. As you can see on the DOMjudge team's interface, it can be divided into multiple components. These regions are organized in a hierarchical tree structure, called the DOM tree, although they are not directly related.

You are a developer of a brand-new browser project. In the project, the elements are modeled as rectangles placed on the Cartesian plane, in such a way that they are properly nested. That is, each pair of rectangles is either strictly contained one another or completely disjoint.

The current algorithm runs in iterations: the browser renders the outermost elements, which are not contained by any others, in a single iteration. These elements are marked as completed and ignored in subsequent iterations. This process repeats until all elements are rendered. Thus, the browser computes *render depth* of an element as one more than the maximum render depth of all the other elements strictly *containing* it, or zero if there is no such element.

However, modern websites often feature more animations, overwhelming the current rendering algorithm. To identify the performance bottleneck, your team needs a tool to analyze the website's structure on the fly. Each element has an animation switch, controlling whether it shows its animation. To get a better view of the page, a key property of an element is whether any portion of it displays its animation. Therefore, an element is called *animated* if it contains some element (including itself) of which the animation switch is on.

To measure how slow an iteration of the rendering process can potentially be, you are given the task of monitoring the resource usage of each iteration. More specifically, you need to handle the toggling of the animation switches, and answer the queries that ask for the number of *animated* elements with some given render depth.

Input

The first line of input contains two integers n ($1 \leq n \leq 5 \times 10^5$) and q ($1 \leq q \leq 5 \times 10^5$), where n is the number of elements and q is the total number of upcoming events.

Each of the next n line contains four integers x_1, y_1, x_2, y_2 ($1 \leq x_1 < x_2 \leq 10^9, 1 \leq y_1 < y_2 \leq 10^9$), indicating the element is a rectangle with bottom-left corner (x_1, y_1) and upper-right corner (x_2, y_2). The boundaries of the rectangles never intersect with each other, not even at a point.

Following this are q lines, indicating the events in chronological order. Each line is either in the format of `^` i, or `?` k. The first format means the animation switch of the i-th element ($1 \leq i \leq n$), in the order of input, should be toggled, i.e. on to off and vice versa. The second format means it is a query of the number of animated elements with render depth k ($0 \leq k$).

Initially, all animation switches are off, and there is at least one event of the second format. The render depth specified in the queries will not exceed the maximum render depth of all elements.

Output

For each query, output the number of animated elements with the given render depth in a line.

Example

standard input	standard output
8 7 1 1 15 11 2 2 7 4 2 5 7 10 3 6 4 9 5 6 6 7 8 2 14 10 9 5 13 9 11 6 12 7 ^ 4 ^ 5 ^ 8 ? 0 ? 1 ? 2 ? 3	1 2 3 1