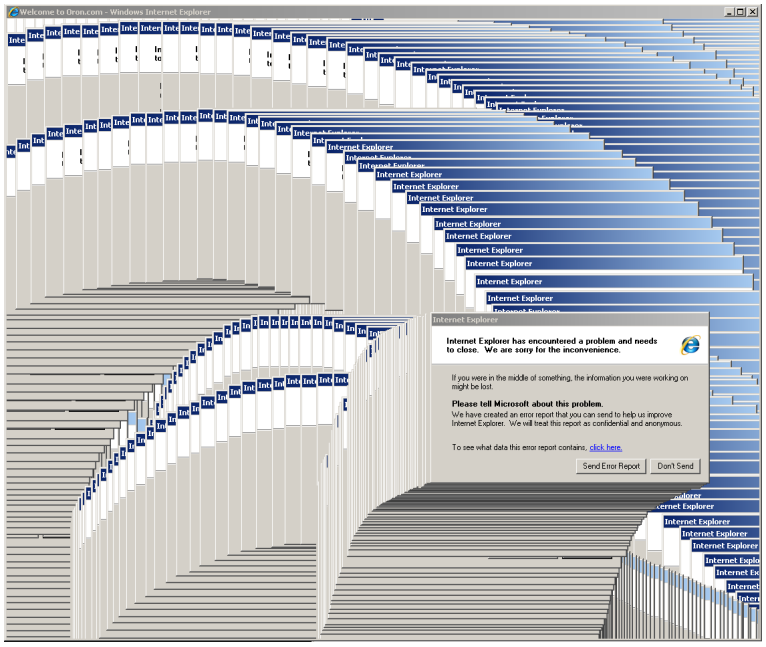
Problem F. Lag

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	1024 mebibytes



You are using Paint on an old Windows computer. The screen of Paint is a grid with cells called pixels. The coordinates of the bottom left pixel are $(1, 1)$, and the coordinates of the pixel that is in a-th column from the left and b-th row from the bottom are (a, b). On the initial screen, N rectangles with vertical and horizontal sides are drawn. A rectangle with bottom left pixel (x_1, y_1) and top right pixel (x_2, y_2) contains all pixels (x, y) such that $x_1 \le x \le x_2$ and $y_1 \le y \le y_2$.

A total of M move commands will be performed on N rectangles. The movement of the rectangle is represented by direction and distance. Each direction is one of the following: east, west, south, north, northeast, northwest, southeast, and southwest (the latter four are 45 degrees to the horizontal axis). Each distance is a positive integer d.

Suppose that the original coordinates of the bottom left pixel of the rectangle are (a, b). A movement by a distance of d in the east, north, west, and south directions causes this pixel to move toward the coordinates $(a + d, b)$, $(a, b + d)$, $(a - d, b)$, and $(a, b - d)$, respectively. In addition, a movement by a distance of d in the northeast, northwest, southwest, and southeast directions causes this pixel to move toward the coordinates $(a + d, b + d)$, $(a - d, b + d)$, $(a - d, b - d)$, and $(a + d, b - d)$, respectively.

Moving by distance d of the rectangle R on the screen is implemented by quickly displaying the shape of R every time when R moves by distance 1. However, our computer is very old, so moving R is very laggy. As a result, all of the R drawn in the movement of R remains on the screen. Therefore, if R moves by the distance d, d rectangles are newly created on the screen. For example, if the rectangle moves in the northeast direction by a distance of 3, 3 rectangles are created, leaving a total of 4 rectangles on the screen. Of course, after moving, the rectangle at the northeast end becomes R.

After executing M move commands, Q queries will be given. Each query is given as a pixel p on the plane. Print the number of rectangles containing the pixel p as the answer to the query.

Input

The first line contains three integers N, M, and Q ($1 \le N \le 250\,000$, $0 \le M \le 250\,000$, $1 \le Q \le 250\,000$).

Each of the next N lines contains four integers x_1, y_1, x_2, and y_2, denoting a rectangle with bottom left pixel (x_1, y_1) and top right pixel (x_2, y_2) ($1 \le x_1 \le x_2 \le 250\,000$, $1 \le y_1 \le y_2 \le 250\,000$).

Each of the next M lines contains three integers v_i, x_i, and d_i, denoting that the x_i-th rectangle moved in the direction v_i by distance d_i ($0 \le v_i \le 7$, $1 \le x_i \le N$, $1 \le d_i \le 250\,000$).

The directions are:

- 0: $(+1, 0)$
- 1: $(+1, +1)$
- 2: $(0, +1)$
- 3: $(-1, +1)$
- 4: $(-1, 0)$
- 5: $(-1, -1)$
- 6: $(0, -1)$
- 7: $(+1, -1)$

Each of the next Q lines contains two integers x and y, denoting the query on pixel (x, y).

All coordinates are positive integers between 1 and 250 000. Any pixels contained in a rectangle at any time satisfy these constraints. Queried pixels also satisfy these constraints.

Output

For each queried pixel, output a single integer denoting the number of rectangles containing the given pixel.

Examples

standard input	*standard output*
1 8 3 2 1 2 1 0 1 1 1 1 1 2 1 1 3 1 1 4 1 1 5 1 1 6 1 1 7 1 1 1 1 2 1 4 2	0 2 1
2 0 3 3 3 7 7 4 4 6 6 5 5 3 7 8 8	2 1 0