Get Jinxed!
Problem ID: getjinxed

Note: This problem uses an encrypted input format to enforce processing queries online. Please pay attention to the input specifiction section for more details.

It's just another day in Jinx's lair. Some of her thousands of monkeys (don't ask) have somehow gotten into her mirror maze (also don't ask). And now—*now!*—she can't find her beloved *Zapper gun* (seriously, **don't ask**). Could one of the monkeys have gotten it? Nah—no chance. Surely. Unless...no, no way. Impossible. ....Right? Well, just in case, Jinx wants to know: if a monkey really did grab the *Zapper* and pulled the trigger inside her maze — who's getting zapped? Can you help her figure it out? (And quickly—before the monkeys get any ideas!)

Jinx's *mirror maze* is built with the following constraints:

- All monkeys and mirrors are exactly at grid points.
- All mirrors point exactly 45° relative to the grid axes; each mirror can be represented as `/` or `\` and both sides of the mirror are reflective.
- The *Zapper* can only be fired parallel to the grid axes.

You are given an initial state of monkeys and mirrors. You are tasked to answer the following types of queries:

- `1 i x y`: Monkey i moves to the coordinates (x, y). It is guaranteed that no monkey or mirror is currently located at (x, y) during this query.
- `2 x y c`: The mirror at (x, y) is toggled, based on c. If c equals `/` or `\`, a mirror corresponding to c is placed at (x, y), replacing any mirror currently at (x, y) (if any exists). If c equals `.` (a period), the mirror at (x, y) is removed without replacement. It is guaranteed if c equals `.` that a mirror is present at (x, y). It is guaranteed no monkey is present at (x, y), regardless of c.
- `3 i d`: Monkey i fires the *Zapper*, in a direction determined by d:
  - If d equals `N`, the monkey fires the *Zapper* north (towards positive y).
  - If d equals `E`, the monkey fires the *Zapper* east (towards positive x).
  - If d equals `S`, the monkey fires the *Zapper* south (towards negative y).
  - If d equals `W`, the monkey fires the *Zapper* west (towards negative x).

The *Zapper* will reflect off mirrors until it hits a monkey or exits the grid. If it would hit a monkey, output that monkey's index for this query. Otherwise, output -1 for this query.

Can you figure out how to save the *Zapper* quickly? . . . Oh, and the monkeys.

Input

On the first line, three integers, N, M, Q ($1 \leq N, M, Q \leq 5 \cdot 10^4$): the number of monkeys, the number of initial mirrors and the number of queries.

- N lines follow, each containing a pair of integers (x_i, y_i) ($0 \leq x_i, y_i \leq 10^6$), the location of the i-th monkey.
- M lines follow, each of the form (x_j, y_j, c). Here, x_j and y_j are integers ($0 \leq x_j, y_j \leq 10^6$), the position of the mirror, and c is either `/` or `\`, representing the type of the mirror.
- Q lines follow, in the query format described above, where
  - For all indices i, $1 \leq i \leq N$
  - For all coordinates x and y, $0 \leq x, y \leq 10^6$

It is guaranteed that the initial placement of monkeys and mirrors contains no duplicate coordinates. It is guaranteed that no two monkeys will ever occupy the same square. It is guaranteed for all queries of type `1`, the destination is empty. It is guaranteed for all queries of type `2`, the target square does not contain a monkey. (One mirror may be replaced by another mirror during the second type of query, however).

To enforce online queries, the x-coordinates and y-coordinates will be encrypted in the following format. Let s_i be the sum of the correct answers to all queries of type 3 before the i-th query, modulo the prime $10^6 + 3$. Then, for any query of type 1 or 2, the coordinates $x'_i \equiv (x_i - s_i) \mod 10^6 + 3$ and $y'_i \equiv (y_i - s_i) \mod 10^6 + 3, 0 \leq x'_i, y'_i < 10^6 + 3$, will be given instead of x_i, y_i. If no such queries of type 3 have occured yet, s_i is 0.

Output

For each query of type 3, output the result of that query on its own line. It is guaranteed there will be at least one such query.

Explanations

For your convenience, the unencrypted sample inputs are given below. Your code will not be tested on the unencrypted inputs.

Sample Input 1 (unencrypted):

```
3 1 9
2 1
2 0
0 2
2 2 \
3 1 N
3 2 N
3 3 E
2 2 2 /
3 1 N
3 2 N
2 2 2 \
3 1 N
3 2 N
```

Sample Input 2 (unencrypted):

```
2 4 17
2 2
3 2
1 1 \
1 3 /
5 1 /
5 3 \
3 1 E
3 1 W
1 1 2 1
3 1 E
3 1 W
1 2 3 3
3 1 E
3 2 W
1 1 1 2
3 2 E
3 2 W
1 1 1 4
3 2 W
2 1 3 \
3 2 W
3 2 E
3 1 S
```

Sample Input 1	**Sample Output 1**
3 1 9 2 1 2 0 0 2 2 2 \ 3 1 N 3 2 N 3 3 E 2 1000000 1000000 / 3 1 N 3 2 N 2 1000000 1000000 \ 3 1 N 3 2 N	3 1 1 -1 1 3 1

Sample Input 2	Sample Output 2
2 4 17 2 2 3 2 1 1 \ 1 3 / 5 1 / 5 3 \ 3 1 E 3 1 W 1 1 1 0 3 1 E 3 1 W 1 2 0 0 3 1 E 3 2 W 1 1 999998 999999 3 2 E 3 2 W 1 1 999996 999999 3 2 W 2 999994 999996 \ 3 2 W 3 2 E 3 1 S	2 -1 1 1 2 1 1 1 2 1 -1 2