

## Stanza degli specchi (specchi)

Mojito wants to get ready for his visit to MUSE, the science museum in Trento, where he's going to challenge Monica in the *hall of mirrors*.

The floor of this room has a rectangular shape and is made of square-shaped tiles which form a grid ( $R$  rows and  $C$  columns). The surface of each tile has some grooves on it that form a “X”. This decoration allows to insert a mirror inside one of the two grooves in order to let it stand straight. We assume that the mirrors are rectangular zero-width panels whose both faces are perfectly reflective.

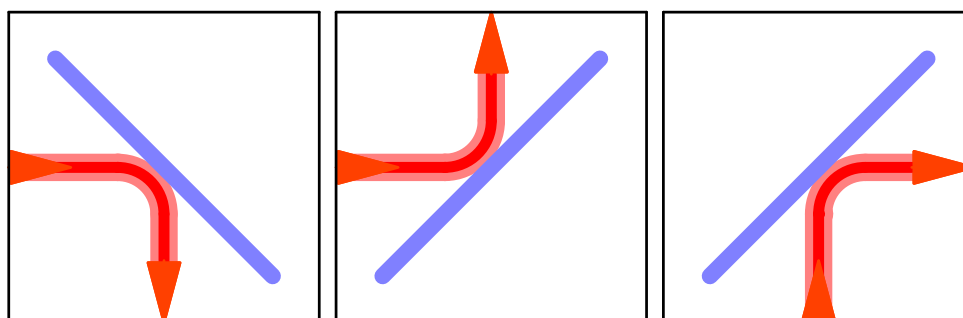


Figure 1: Some ways in which light can be reflected

There are holes on each of the four walls of the room, precisely above each row and above each column of floor tiles. These holes allow Monica to point a laser beam. Once the beam is in the room, it *bounces* on every mirror it encounters on his trip (possibly none) and, eventually, it leaves the room from one of the other holes in the walls. That is where Mojito would like to be, so that he can catch the beam that is going out.

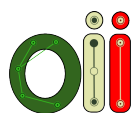
The room is initially empty, and the tiles don't have any mirror on them. During the game, Monica can choose to add a mirror on one of the free tiles (the ones without any mirror on them), choosing also which diagonal to put the mirror on. Alternatively, she can choose to shoot a laser beam through a hole. Each time Mojito notices that Monica is about to shoot a laser in some hole, he tries to guess the exact exiting hole of the laser beam.

Help Mojito by writing a program to find the exiting hole, keeping track of the mirrors added by Monica during the game!

### Clarifications


The tile grid is made of  $R$  rows, numbered from left to right with indexes 0 to  $R - 1$  and  $C$  columns numbered from top to bottom with indexes 0 to  $C - 1$ . The four walls of the room are named SOPRA (top), DESTRA (right), SOTTO (bottom), SINISTRA (left). There are  $R$  holes on the DESTRA and SINISTRA walls and  $C$  holes on the SOPRA and SOTTO walls. In total there are  $2R + 2C$  holes. The holes on each wall are identified by the row or column number on which they are located.

It's *not allowed* to put two mirrors on the same tile, even on different diagonals. The mirrors are located *exactly* along the diagonals, so the laser beam keeps parallel to the room walls and goes along the direction of either the rows or the columns. The width of the mirrors is negligible, so even after all the bounces the laser beam line will still be inside the area of the exiting hole.



## Implementation

You need to submit only one file, with extension `.cpp` o `.c`.

 You can find as an attachment to this task a template `mirrors.cpp` and `mirrors.c` which is an example implementation.

You will need to implement the following:

```
C/C++ void inizia(int R, int C);
```

The function `inizia` is called with the following values of the parameters, at the beginning:

- the integer number  $R$  that represents the number of rows in the room,
- the integer number  $C$  that represents the number of column in the room.

```
C/C++ void aggiungi(int r, int c, char diagonale);
```

The function `aggiungi` is called everytime Monica adds a new mirror, with the following values as parameters:

- an integer number  $r$ , in a range between 0 and  $R - 1$ , which indicates the index of the row in which the mirror is added,
- an integer number  $c$ , in a range between 0 and  $C - 1$ , which indicates the index of the column in which the mirror is added,
- a character `diagonale`, which indicates the orientation of the mirror and that can only have the following values: `'\'` or `'/'`.

```
C/C++ foro_t calcola(foro_t ingresso);
```

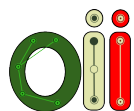
The function `calcola` is called everytime Monica shoots a laser beam in a hole.

- The parameter `ingresso`, of type `foro_t`, indicates the “entering” hole (the one Monica shoots the beam in).
- The function returns the “exiting” hole (the one the laser beam exits through, stored in a data structure `struct` of type `foro_t`).

The data type `foro_t` is a `struct` that indicates a hole on a particular wall and it contains the following fields:

- `parete`: a `enum` that may have the values `SOPRA`, `DESTRA`, `SOTTO` or `SINISTRA` and defines the wall of the hole,
- `posizione`: an integer number in a range from 0 to  $R - 1$  (if `parete` has the value `DESTRA` or `SINISTRA`) or between 0 and  $C - 1$  (if `parete` has value `SOPRA` or `SOTTO`), which indicates the row or column of the hole.

The function `inizia` is called first, exactly once. Later `aggiungi` or `calcola` are called, without any particular ordering. The functions `aggiungi` or `calcola` are called exactly  $Q$  times. The number  $Q$  isn't known in advance to the program, but it follows the assumptions of the following sections.



## Sample Grader

You can find a simplified version of the grader used during the competition in this task's directory. You can use it to test your solutions. The sample grader reads data from `stdin`, calls the functions that you have to implement and writes on `stdout`, according to the following format.

The input file consists of  $Q + 1$  rows:

- line 1: the integer  $R$ ,  $C$  and  $Q$ .
- line  $i$ , for  $i = 2, 3, \dots, Q + 1$ : a character  $c$  followed by two integer numbers  $a$  and  $b$ .
  - If the character  $c$  is `?`, the function `calcola` is called: the first number  $a$  represents the wall from 0 to 3, clockwise starting from `SOPRA`) and the second number  $b$  represents the position of the hole.
  - If the character  $c$  is `'\'` or `'/'`, the function `inserisci` is called: the number  $a$  represents the index of the row and  $b$  represents the index of the column.

The output file consists of a row for each call to the function `calcola`, which contains the return value of that call.

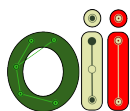
## Constraints

- $1 \leq R, C \leq 10\,000$ .
- $1 \leq Q \leq 1\,000\,000$
- No mirror will be inserted on a tile that already has a mirror on it.

## Scoring

Your code will be tested on several testcases grouped in subtasks. To get the score of a subtask it's necessary to solve completely all the tests the subtask is made of.

- **Subtask 1** [ 0 points]: Sample cases.
- **Subtask 2** [ 8 points]:  $R, C \leq 10$  and  $Q \leq 100$ .
- **Subtask 3** [21 points]:  $R, C \leq 1000$  and  $Q \leq 5000$ .
- **Subtask 4** [14 points]:  $R, C \leq 1000$ ,  $Q \leq 100\,000$  and all calls to `aggiungi` precede all the calls to `prevedi`.
- **Subtask 5** [24 points]:  $R, C \leq 100\,000$  and  $Q \leq 5000$ .
- **Subtask 6** [13 points]:  $R, C \leq 100\,000$ ,  $Q \leq 250\,000$  and all calls to `aggiungi` precede all the calls to `prevedi`.
- **Subtask 7** [12 points]:  $R, C \leq 100\,000$  e  $Q \leq 100\,000$ .
- **Subtask 8** [ 8 points]: No specific limitations.

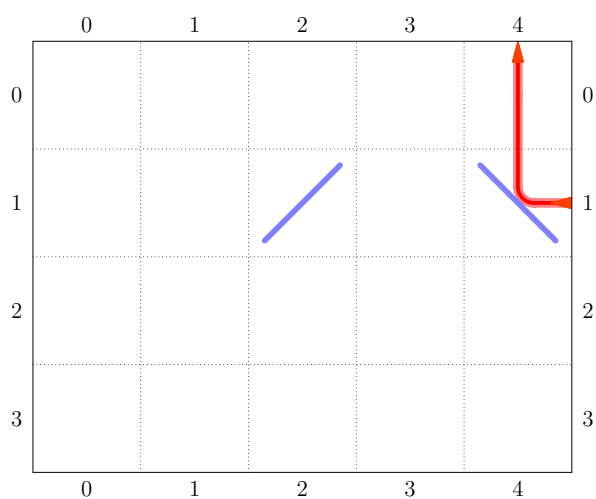


## Examples

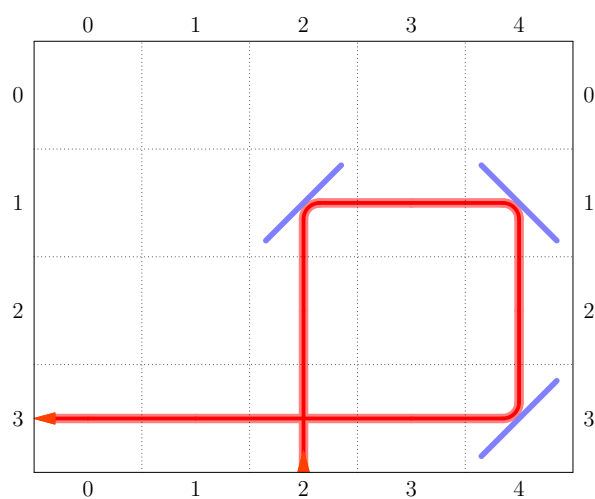
stdin	stdout
4 5 5 / 1 2 \ 1 4 ? 1 1 / 3 4 ? 2 2	0 4 3 3
6 2 7 ? 1 1 / 1 1 \ 4 0 / 4 1 / 1 0 ? 0 1 ? 1 1	3 1 1 1 0 1

## Explanation

The requirements of the **first example case** may be represented this way:



$\text{prevedi}(\{ \text{DESTRA}, 2 \}) \Rightarrow \{ \text{SOPRA}, 5 \}$



$\text{prevedi}(\{ \text{SINISTRA}, 4 \}) \Rightarrow \{ \text{SOTTO}, 3 \}$

The **second example case** is the following:

