Stanza degli specchi (`specchi`)

Mojito intende prepararsi per la sua visita al MUSE, il museo delle scienze di Trento, dove lo attende una sfida con Monica nella *stanza degli specchi*.

Il pavimento della stanza è rettangolare, ed è ricoperto con piastrelle quadrate disposte su una griglia di R righe e C colonne. Ogni piastrella presenta due scanalature diagonali a "X" che consentono l'eventuale inserimento di uno specchio posizionato in piedi lungo una delle due diagonali della piastrella. Gli specchi sono dei pannelli rettangolari senza spessore, con entrambe le facce perfettamente riflettenti.

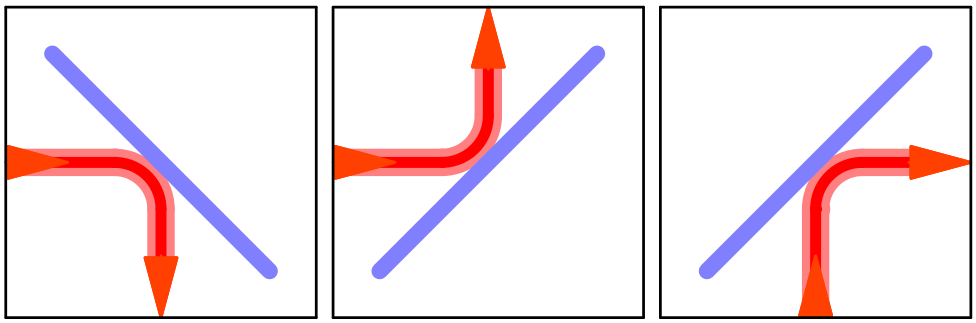


Figura 1: Alcuni dei possibili modi in cui la luce si può riflettere

Su ciascuna delle quattro pareti della stanza, in corrispondenza di ogni riga o colonna, è presente un foro attraverso il quale Monica può immettere un raggio laser. Una volta entrato nella stanza, il raggio *rimbalza* sugli specchi che incontra lungo il proprio percorso (possibilmente nessuno), e infine fuoriesce da uno degli altri fori. È lì che Mojito vorrebbe farsi trovare pronto ad accogliere ed acchiappare il raggio uscente.

All'inizio la stanza è vuota, ossia tutte le piastrelle sono prive di specchi. Nel corso del gioco, Monica può decidere di aggiungere uno specchio su una delle piastrelle ancora libere, posizionandolo su una delle due diagonali a sua scelta, oppure può immettere il raggio laser attraverso un foro. Ogni volta che Mojito vede Monica avvicinarsi ad uno dei fori, cerca di anticipare da quale foro fuoriuscirà il laser.

Aiuta Mojito scrivendo un programma che calcoli il foro di uscita del raggio, tenendo conto degli specchi man mano aggiunti da Monica!

Specifiche

Le griglia di piastrelle è formata da R righe, numerate da sinistra verso destra con i numeri da 0 a $R-1$, e C colonne numerate dall'alto verso il basso con i numeri da 0 a $C-1$. Le quattro pareti della stanza sono chiamate `SOPRA`, `DESTRA`, `SOTTO`, `SINISTRA`. Vi sono R fori sulle pareti `DESTRA` e `SINISTRA`, e C fori sulle pareti `SOPRA` e `SOTTO`, per un totale di $2R+2C$ fori. I fori su ciascuna parete sono identificati dal numero della riga o colonna su cui si affacciano.

Non possono *mai* trovarsi due specchi sulla stessa piastrella, nemmeno su diagonali diverse. Inoltre, gli specchi sono posizionati in modo molto preciso lungo le diagonali, perciò il raggio laser rimane parallelo alle pareti della stanza, e procede sempre esattamente lungo una delle righe o delle colonne. Lo spessore degli specchi è trascurabile, anche dopo tutti i rimbalzi la linea del raggio non fuoriesce dall'area dei fori.

Implementazione

Dovrai sottoporre un unico file, con estensione `.cpp` o `.c`.

> ☞ Tra gli allegati a questo task troverai un template `specchi.cpp` e `specchi.c` con un esempio di implementazione.

Dovrai implementare le seguenti funzioni:

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

La funzione `inizia` viene invocata all'inizio con i seguenti parametri:

- l'intero R che rappresenta il numero di righe della stanza,
- l'intero C che rappresenta il numero di colonne della stanza.

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

La function `aggiungi` viene chiamata ogni volta che Monica aggiunge uno specchio, con i seguenti parametri:

- un numero intero `r`, compreso fra 0 a $R-1$, che indica il numero della riga in cui viene aggiunto lo specchio,
- un numero intero `c`, compreso fra 0 a $C-1$, che indica il numero della colonna in cui viene aggiunto lo specchio,
- un carattere `diagonale` che indica l'orientamento dello specchio e che può assumere solo i valori '\' o '/'.

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

La funzione `calcola` viene chiamata ogni volta che Monica introduce il raggio laser in un foro.

- Il parametro `ingresso`, di tipo `foro_t`, indica il foro di ingresso del raggio.
- La funzione deve restituire il foro di uscita del raggio, in una `struct` di tipo `foro_t`.

Il tipo di dato `foro_t` è una `struct` che specifica un foro su una parete della stanza, e contiene i seguenti campi:

- `parete`: un `enum` che può assumere i valori `SOPRA`, `DESTRA`, `SOTTO` o `SINISTRA` e che individua la parete su cui si trova il foro,
- `posizione`: un intero compreso tra 0 e $R-1$ (se `parete` è `DESTRA` o `SINISTRA`) o tra 0 e $C-1$ (se `parete` è `SOPRA` o `SOTTO`), che indica la riga o colonna su cui si affaccia il foro.

La funzione `inizia` sarà chiamata per prima, una sola volta. Successivamente verranno chiamate le funzioni `aggiungi` o `calcola`, in nessun ordine specifico. Le funzioni `aggiungi` o `calcola` vengono chiamate per un totale di Q volte. Il numero Q non è specificato in anticipo al programma, ma rispetta le assunzioni indicate nelle sezioni successive.

Grader di prova

Nella directory relativa a questo problema è presente una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama le funzioni che dovete implementare e scrive su `stdout`, secondo il seguente formato.

Il file di input è composto $Q + 1$ righe, contenenti:

- Riga 1: gli interi R, C e Q.
- Righe $2, \ldots, Q + 1$: un carattere c seguito da due numeri interi a e b.
  - Se il carattere c è `?` viene chiamata la funzione `calcola`, dove il primo numero a indica la parete (da 0 a 3 in senso orario: `SOPRA`, `DESTRA`, `SOTTO`, `SINISTRA`) e il secondo numero b indica la posizione del foro.
  - Se il carattere c è `'\'` o `'/'`, viene chiamata la funzione `aggiungi` dove il numero a indica la riga e il numero b la colonna.

Il file di output contiene una riga per ogni chiamata alla funzione `calcola`, contenente il valore di ritorno di tale chiamata.

Assunzioni

- $1 \leq R, C \leq 100\,000$.
- $1 \leq Q \leq 250\,000$
- Non viene mai inserito più di uno specchio nella stessa posizione.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test che lo compongono.

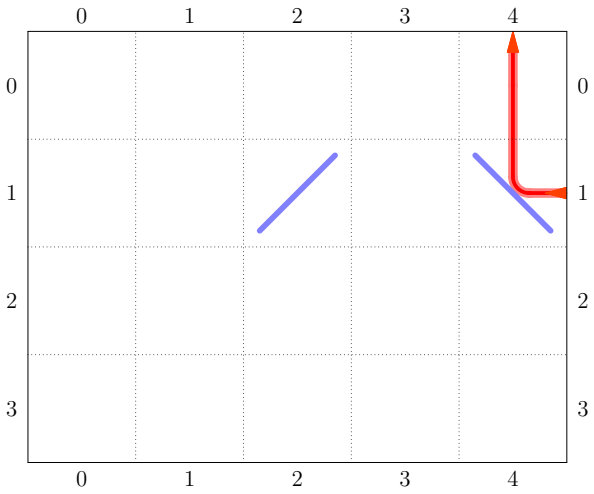
- **Subtask 1 [0 punti]**: Casi d'esempio.
- **Subtask 2 [8 punti]**: $R, C \leq 10$ e $Q \leq 100$.
- **Subtask 3 [21 punti]**: $R, C \leq 1000$ e $Q \leq 5000$.
- **Subtask 4 [14 punti]**: $R, C \leq 1000$, $Q \leq 100\,000$ e tutte le chiamate a `aggiungi` precedono tutte le chiamate a `calcola`.
- **Subtask 5 [24 punti]**: $R, C \leq 100\,000$ e $Q \leq 5000$.
- **Subtask 6 [13 punti]**: $R, C \leq 100\,000$, $Q \leq 250\,000$ e tutte le chiamate a `aggiungi` precedono tutte le chiamate a `calcola`.
- **Subtask 7 [12 punti]**: $R, C \leq 100\,000$ e $Q \leq 100\,000$.
- **Subtask 8 [8 punti]**: Nessuna limitazione specifica.

Esempi di input/output

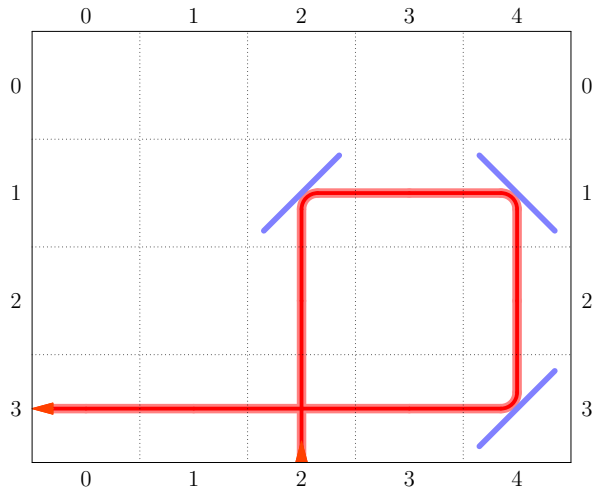
stdin	stdout
4 5 5 / 1 2 \ 1 4 ? 1 1 / 3 4 ? 2 2	0 4 3 3
6 2 7 ? 1 1 / 1 1 \ 4 0 / 4 1 / 1 0 ? 0 1 ? 1 1	3 1 1 1 0 1

Spiegazione

Le richieste del **primo caso d'esempio** possono essere così rappresentate:

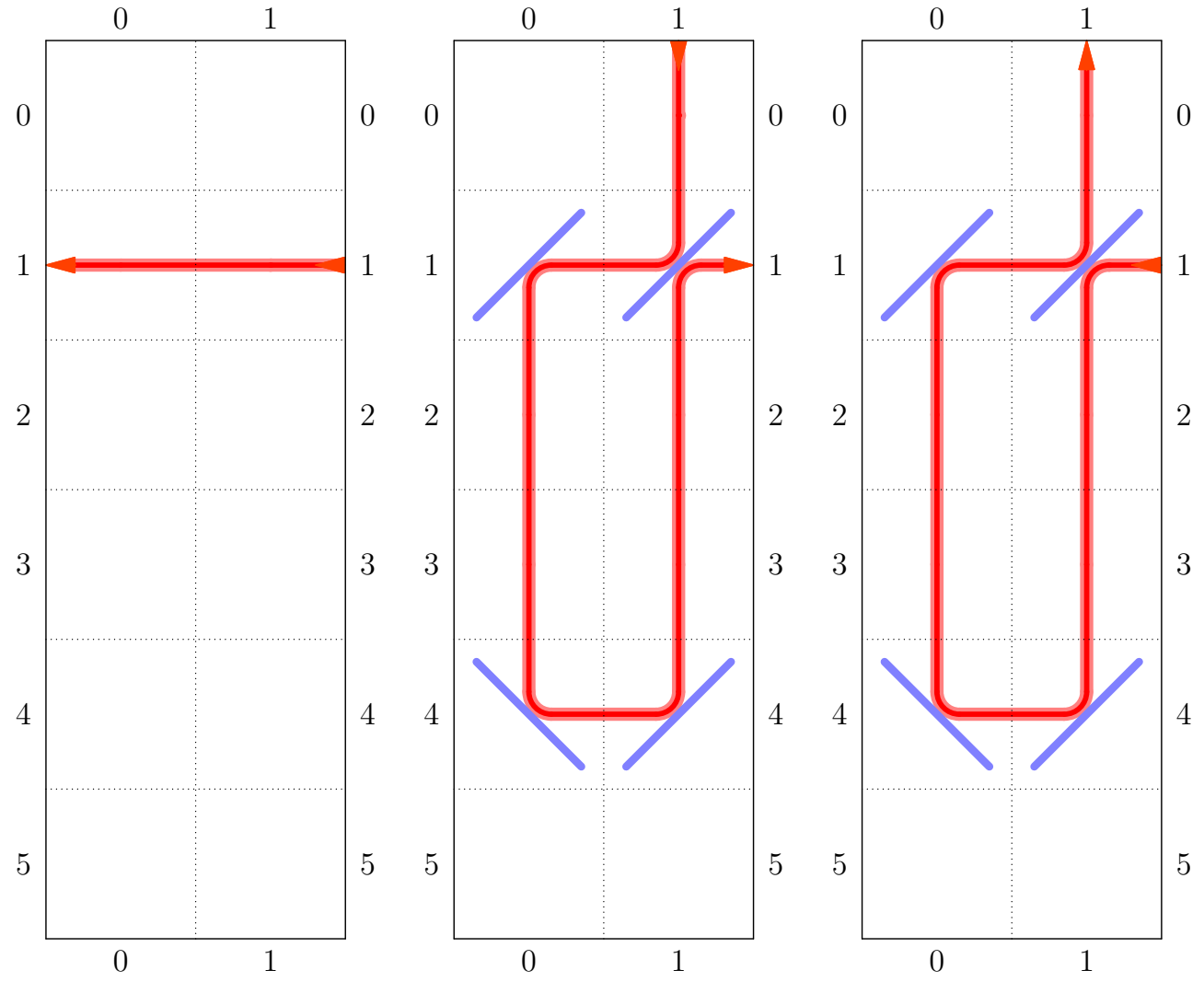


`calcola({ DESTRA, 1 }) ⇒ { SOPRA, 4 }`



`calcola({ SOTTO, 2 }) ⇒ { SINISTRA, 3 }`

Il **secondo caso d'esempio** è il seguente:



Visualizzatore

All'indirizzo `https://contest.oii2017.tk/vis_specchi/` è presente una pagina web con un visualizzatore del problema. Inserendo il file di input è possibile simulare le diverse query.