Un `TiltMaze` realizza una versione digitale del classico gioco del labirinto con pallina bilanciata.

Il gioco è costituito da una griglia di R x C caselle. Ognuna di esse può essere vuota, occupata da un muro, da una buca, oppure rappresentare il punto di start o il punto di exit.

Sulla griglia si muove una pallina che simula le leggi fisiche di un piano inclinato: una volta iniziata la scivolata in una direzione, la pallina si ferma solo se incontra un muro o i bordi del quadro. Se la pallina finisce in una buca, la partita è persa. Se raggiunge il punto di exit, la partita è vinta. Prima di avviare il gioco, sarà necessario costruire il labirinto. Le operazioni di modifica dello schema del labirinto avvengono prima dell'avvio del gioco. Indicare lo stato del gioco usando una variabile enumeratore di tipo:

```
enum Stato{MODIFICA, AVVIATO, VINTO, PERSO};
```

Implementare le seguenti operazioni che possono essere effettuate su un `TiltMaze`.

--- Metodi invocati nella PRIMA PARTE di main.cpp: ---

✓ **`TiltMaze t(R, C, rSt, cSt, rEx, cEx);`**

Costruttore che inizializza un labirinto vuoto di dimensioni `R x C`, se sia `R` che `C` sono validi. Se anche solo uno dei due non lo è, il labirinto viene inizializzato con `R=3` e `C=3`.

Vengono definiti il punto di start (`rSt, cSt`) e il punto di exit (`rEx, cEx`).

Inizialmente il gioco si trova nello stato `MODIFICA`, la pallina non è presente sulla griglia e non ci sono né muri né buche (ad eccezione dei bordi esterni impliciti che delimitano l'area di gioco).

Le coordinate dei punti di start e exit, se fuori dalla griglia, vengono bloccate tra `0` e `R-1` (per gli indici di riga) e `0` e `C-1` (per gli indici di colonna). In altre parole, se `rSt` o `rEx` sono minori di `0`, vengono settati a `0`, mentre se sono maggiori di `R-1` vengono settati a `R-1`; allo stesso modo, se `cSt` o `cEx` sono minori di `0` vengono settati a `0`, mentre se sono maggiori di `C-1` vengono settati a `C-1`.

✓ **`t.aggiungiMuro(r1, c1, r2, c2);`**

Crea un segmento di muro tra le coordinate (`r1, c1`) e (`r2, c2`), ritornando `true` se l'operazione ha successo.

Il muro può essere solo orizzontale (stessa riga) o verticale (stessa colonna). Tutte le celle comprese tra le due coordinate (estremi inclusi) diventano muri. Inoltre, la coordinata `r2` deve essere sempre maggiore o uguale a `r1` quando si definisce una linea verticale; allo stesso modo, la coordinata `c2` deve essere sempre maggiore o uguale a `c1` quando si definisce una linea orizzontale.

Se le coordinate non individuano un segmento orizzontale o verticale, se non rispettano il vincolo che `r2>=r1` e `c2>=c1`, se escono dall'area di gioco, o se vanno a coprire il punto di start o di exit, la funzione restituisce `false` e non modifica il labirinto.

L'operazione è consentita solo se lo stato è `MODIFICA`, altrimenti nuovamente la funzione restituirà `false` senza modificare lo stato del labirinto.

✓ **`t.aggiungiBuca(r, c);`**

Inserisce una buca nella coordinata specificata, ritornando `true` se l'operazione ha successo. Se nella cella era presente un muro, esso viene rimosso e sostituito dalla buca.

Non è possibile piazzare una buca sul punto di start o di exit e l'operazione è consentita solo se lo stato è `MODIFICA`. Alternativamente, la funzione restituisce `false` lasciando il labirinto inalterato.

✓ **`t.avvia();`**

Metodo che avvia (o riavvia nel caso in cui la partita fosse stata precedentemente vinta o persa) la partita. La pallina viene posizionata esattamente sulle coordinate del punto di start (`rSt, cSt`).

Lo stato del gioco diventa `AVVIATO`.

✓ **`cout << t;`**

Operatore di uscita per il tipo `TiltMaze`. Deve prima stampare lo stato del gioco nel formato "Stato: STATO", seguito da un accapo, e poi disegnare la griglia.

I caratteri # rappresentano i muri, `O` (o maiuscola) le buche, `S` il punto di start, `E` il punto di exit.

La pallina, se presente (gioco avviato), viene indicata con il carattere `x`, che deve sempre essere visibile quando il gioco è in stato avviato (la `x` ha la priorità sugli altri caratteri). Se il gioco viene perso o vinto, o è in modalità modifica, la pallina non viene stampata.

Le coordinate sono stampate a lato e in basso, a partire da 0. Inoltre, si noti che ogni carattere è separato con uno spazio dal successivo.

Esempio di output in fase di modifica:	Esempio di output in fase di gioco avviato

`Stato: MODIFICA`
`4 S . . . .`
`3 . . # # .`
`2 . O . . .`
`1 . . . . .`
`0 . . . . E`
`  0 1 2 3 4`
	`Stato: AVVIATO`
	`4 x . . . .`
	`3 . . # # .`
	`2 . O . . .`
	`1 . . . . .`
	`0 . . . . E`
	`  0 1 2 3 4`

✓ **`t.stato();`**

Restituisce lo `Stato` corrente del gioco, come letterale enumerato (`MODIFICA, AVVIATO, VINTO, PERSO`).

--- Metodi invocati nella SECONDA PARTE di main.cpp: ---

✓ **`~TiltMaze();`**

Ridefinire il distruttore, nel caso sia necessario farlo.

✓ **`t.inclina(dirV, dirH);`**

Inclina il piano di gioco per far scivolare la pallina.

I parametri indicano l'inclinazione:

- dirV: +1 (Inclinato verso l'alto, la pallina va su), -1 (Verso il basso), 0 (Nessuna pendenza verticale).
- dirH: +1 (Inclinato verso destra), -1 (Verso sinistra), 0 (Nessuna pendenza orizzontale).

È ammessa una sola inclinazione alla volta (es: inclina(1, 0) o inclina(0, -1)). Se vengono specificati entrambi i valori diversi da 0, o entrambi 0, la funzione **non ha effetto**.

Fisica del movimento: La pallina "scivola" nella direzione indicata finché non accade una delle seguenti condizioni:

- Incontra un muro o il bordo del labirinto: si ferma nella cella precedente l'ostacolo.
- Passa sopra una buca: la pallina cade, sparisce dalla griglia e lo stato diventa `PERSO`.
- Incontra il punto di uscita: lo stato diventa `VINTO`.

La funzione `inclina` non ha effetto se non si è in stato `AVVIATO`.

Nota Bene: il punto di start, una volta avviato il gioco, viene considerato come uno spazio vuoto della griglia (non è un muro e non cambia lo stato del gioco se attraversato).

Esempio di sequenza di scivolata:

Inizialmente:	Inclina verso destra (0, +1)	Inclina verso il basso (-1, 0)
`Stato: AVVIATO`	`Stato: AVVIATO`	`Stato: VINTO`
`x . . . . #`	`. . . . x #`	`. . . . . #`
`S . . . E .`	`S . . . E .`	`S . . . E .`

✓ **`TiltMaze t2(t1);`**

Costruttore di copia. Crea un nuovo labirinto `t2` identico a `t1` per dimensioni e disposizione di muri/buche/start/exit.

Nota Bene: Indipendentemente dallo stato di `t1`, il nuovo labirinto `t2` viene inizializzato **sempre** in stato `MODIFICA` e **senza la pallina in gioco**.

✓ **`t1 + t2;`**

Operatore di somma che concatena due labirinti orizzontalmente e restituisce il nuovo labirinto risultante. L'operazione è valida solo se i due labirinti hanno lo stesso numero di righe. In caso contrario, viene restituita una copia di `t1`. Il labirinto risultante è costruito come segue:

Il labirinto `t1` viene posizionato a sinistra.

Il labirinto `t2` viene posizionato a destra.

Il punto di start (S) del nuovo labirinto è quello di `t1`.

Il punto di exit (E) del nuovo labirinto è quello di `t2`.

Il labirinto risultante è inizializzato in modalità `MODIFICA` (pallina assente).

Mediante il linguaggio C++, realizzare il tipo di dato astratto **`TiltMaze`**, definito dalle precedenti specifiche. **Gestire le eventuali situazioni di errore**. Non è permesso utilizzare funzionalità della libreria STL (Standard Template Library) come il tipo `string`, il tipo `vector`, il tipo `list`, ecc. Ovviamente le librerie `cmath`, `cstdlib` e `cstring` (`strlen`, `strcpy`, `strcat`, ...) possono essere utilizzate tranquillamente come sempre.

Note per la consegna

Affinché l'elaborato venga considerato valido, il programma **deve** produrre almeno la prima parte dell'output atteso. In questo caso, i docenti procederanno alla valutazione dell'elaborato **solo se** lo studente avrà completato l'autocorrezione del proprio elaborato.

In **tutti** gli altri casi (per esempio, il programma non compila, non collega, non esegue o la prima parte dell'output non coincide con quella attesa), l'elaborato è considerato **insufficiente** e, pertanto, **non verrà corretto**.

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---
Gioco in modifica? true
Stato: MODIFICA
3 . . . .
2 . . . .
1 . . . .
0 S . . E
  0 1 2 3
Stato: MODIFICA
3 O # # #
2 . . . .
1 . . # .
0 S . # E
  0 1 2 3
Gioco avviato? true
Stato: AVVIATO
3 O # # #
2 . . . .
1 . . # .
0 x . # E
  0 1 2 3

--- SECONDA PARTE ---
Stato: AVVIATO
3 O # # #
2 . . . .
1 . . # .
0 S x # E
  0 1 2 3
Stato: AVVIATO
3 O # # #
2 . x . .
1 . . # .
0 S . # E
  0 1 2 3
Stato: AVVIATO
3 O # # #
2 . . . x
1 . . # .
0 S . # E
  0 1 2 3
Stato: VINTO
3 O # # #
2 . . . .
1 . . # .
0 S . # E
  0 1 2 3
Gioco vinto? true
Stato: PERSO
3 O # # #
2 . . . .
1 . . # .
0 S . # E
  0 1 2 3
Stato: MODIFICA
3 O # # # . . . . . E
2 . . . . . .
1 . . # O . . . . . .
0 S . # . . . O . . .
  0 1 2 3 4 5 6 7 8 9
```