Un `MarioStage` realizza un livello di una versione semplificata del celebre videogioco Super Mario Bros®. Un `MarioStage` è formato da una griglia di 8x32 caselle, ognuna delle quali può essere vuota, occupata da un blocco, oppure occupata da Mario o da una Koopa (una tartaruga nemica di Mario). Un livello può contenere al più una Koopa. Ogni casella è identificata da una coppia di indici (*i,j*), con *i* indice di riga da 0 a 7, e *j* indice di colonna da 0 a 31. Un `MarioStage` può essere nello stato "gioco avviato" o "gioco fermo". L'obiettivo del gioco è far arrivare Mario indenne nell'ultima colonna a destra (quella con indice 31). Mario può morire cadendo nella lava presente alla riga inferiore (quella con indice 0), oppure (vedi seconda parte) se ucciso dalla Koopa. Sia nel caso in cui Mario vinca che in quello in cui muoia, il gioco si ferma.

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

--- PRIMA PARTE ---

✓ `MarioStage stage;`
Costruttore di default che inizializza un `MarioStage stage` vuoto, in cui cioè non ci sono blocchi. All'inizio, il gioco è fermo.

✓ `cout << t;`
Operatore di uscita per il tipo `MarioStage`. L'uscita (nel caso di gioco avviato) ha la forma seguente:

```
7
6
5                 =====
4 M
3========
2============  =====
1============  =====
0============  ==================
 01234567890123456789012345678901
```

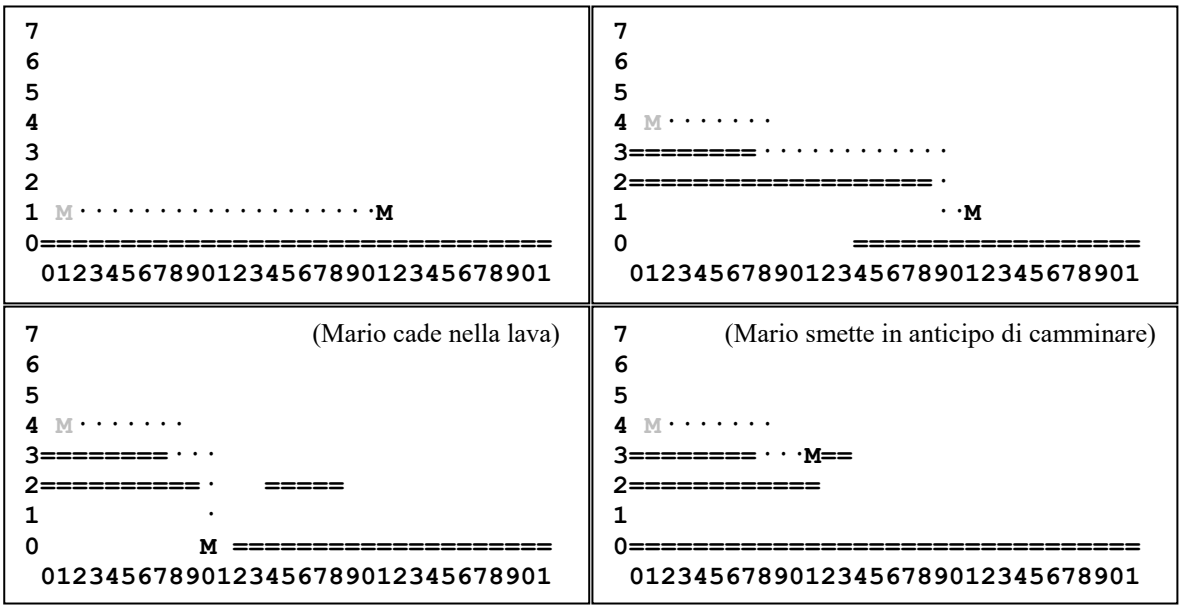
dove i numeri sul lato sinistro rappresentano gli indici di riga, e quelli sul lato inferiore rappresentano l'ultima cifra decimale degli indici di colonna. Notare lo spazio nell'angolo in basso a sinistra, tra gli indici di riga e quelli di colonna. I caratteri '**=**' rappresentano i blocchi e il carattere '**M**' rappresenta la posizione attuale di Mario. Il disegno deve essere seguito da un singolo accapo finale. **ATTENZIONE:** tale disegno viene stampato ***solo*** in caso di gioco avviato. Se invece il gioco è fermo, viene semplicemente stampata la scritta maiuscola "**(NOT PLAYING)**" seguita da un singolo accapo.

✓ `stage.drawBlocks(i1,j1,i2,j2);`
Operazione che disegna un rettangolo di blocchi sul livello `stage`. Il rettangolo di blocchi ha come limite inferiore-sinistro la casella (*i1,j1*) e come limite superiore-destro la casella (*i2,j2*). La funzione può essere invocata solo a gioco fermo, altrimenti essa non ha effetto. Deve essere possibile concatenare le chiamate a `drawBlocks`, per esempio: `stage.drawBlocks(1,2,3,4).drawBlocks(5,6,7,8).`

✓ `stage.play(i,j);`
Operazione che avvia (o ri-avvia) il gioco sul livello `stage`. Mario viene collocato nella casella di partenza (*i,j*). Tale casella di partenza deve essere una casella vuota immediatamente sopra un blocco, e non nella riga in basso (dove c'è la lava) né nell'ultima colonna (dove Mario vince). Se tali condizioni non si verificano, la funzione non ha effetto.

✓ `stage.walkMario(n);`

Operazione che fa camminare Mario in avanti (cioè verso destra nel livello) di `n` caselle. Se durante la camminata Mario non trova più un blocco sotto ai piedi, egli cade in verticale verso il basso finché non atterra su un blocco, per poi continuare a camminare. Mario può anche cadere nella lava alla riga di indice 0, in tal caso il gioco si ferma. Se durante la camminata Mario trova un blocco davanti a sé, egli smette in anticipo di camminare. Seguono quattro esempi di camminata, tutti di `n=20` caselle. La '**M**' grigia rappresenta la posizione iniziale di Mario, quella nera la posizione finale.



Deve essere possibile concatenare le chiamate a `walkMario`. La funzione può essere invocata solo a gioco avviato, altrimenti essa non ha effetto.

--- SECONDA PARTE ---

✓ `stage.setKoopa(i,j);`

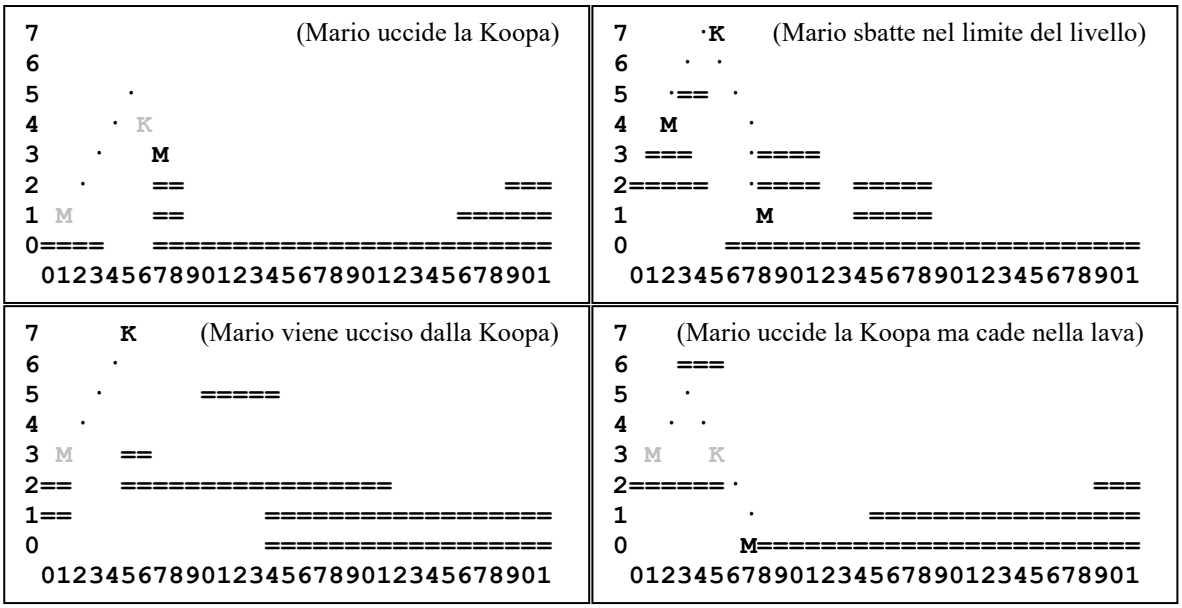
Operazione che assegna la posizione della Koopa nel livello `stage`. In ogni livello può essere presente al più una sola Koopa. L'operatore di uscita deve rappresentare la posizione della Koopa nel livello con un carattere '**K**'. La funzione può essere invocata solo a gioco fermo, e la posizione della Koopa deve essere una casella vuota. Se tali condizioni non si verificano, la funzione non ha effetto. Non è necessario che la Koopa abbia un blocco sotto ai piedi: essa può fluttuare nell'aria. La Koopa resta immobile durante il gioco.

✓ `stage.walkMario(n);` (MODIFICA)

Modificare la funzione `walkMario` per tenere conto dell'eventuale Koopa. Se Mario collide con la Koopa, ne è ucciso se stava camminando, e invece la uccide se stava cadendo. La Koopa uccisa deve sparire dal livello, per poi riapparire nella stessa posizione al ri-avvio del gioco.

✓ `stage.jumpMario(n);`

Operazione che fa saltare Mario in avanti per un'altezza di `n` caselle. Il salto è composto da una fase ascendente in cui Mario sale, cioè si sposta verso destra e in su, e una fase discendente in cui Mario scende, cioè si sposta verso destra in giù. La fase ascendente dura per `n` caselle, o fino a quando Mario non può più salire. La fase discendente dura finché Mario non trova un blocco sotto ai piedi, o cade nella lava. Se durante il salto il movimento di Mario è ostacolato da un blocco, egli smette temporaneamente di avanzare verso destra, e quindi si sposta in su (se in fase ascendente) o in giù (se in fase discendente). Se Mario collide con la Koopa, ne è ucciso se era in fase ascendente, e invece la uccide se era in fase discendente. Seguono quattro esempi di salto, tutti di `n=4` caselle.



Deve essere possibile concatenare le chiamate a `jumpMario`. La funzione può essere invocata solo a gioco avviato, altrimenti essa non ha effetto.

Mediante il linguaggio C++, realizzare il tipo di dato astratto definito dalle precedenti specifiche. Non è permesso utilizzare funzionalità della libreria STL come il tipo `string`, il tipo `vector`, il tipo `list`, ecc.