Un `SonicLevel` realizza un livello di una versione semplificata del celebre videogioco Sonic the Hedgehog®. Un `SonicLevel` è formato da una griglia di 8x32 caselle, ognuna delle quali può essere vuota, occupata da un blocco, occupata da un anello (un anello d'oro che protegge Sonic dai pericoli), occupata da uno spuntone affilato che può danneggiare Sonic, oppure occupata da Sonic. 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 `SonicLevel` può essere nello stato "gioco avviato" o "gioco fermo". L'obiettivo del gioco è far arrivare Sonic indenne nell'ultima colonna a destra (quella con indice 31). Sia nel caso in cui Sonic vinca che in quello in cui muoia, il gioco si ferma.

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

--- PRIMA PARTE ---

✓ `SonicLevel s;`
Costruttore di default che inizializza un `SonicLevel` di base, in cui cioè tutte le caselle sono vuote fuorché quelle della riga in basso, che sono riempite di blocchi. All'inizio, il gioco è fermo.

✓ `cout << s;`
Operatore di uscita per il tipo `SonicLevel`. L'uscita (nel caso di gioco avviato) ha la forma seguente:

```
Anelli:4
7
6
5         o  o       =====
4 S
3========o      ooo
2==================      oooo
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. Il numero di anelli raccolti da Sonic è stampato all'inizio preceduto dalla scritta "**`Anelli:`**". I caratteri '**`=`**' rappresentano i blocchi, i caratteri '**`o`**' minuscoli gli anelli, e il carattere '**`S`**' maiuscolo la posizione attuale di Sonic. 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 "**`(GIOCO FERMO)`**" seguita da un singolo accapo.

✓ `s.blocchi(i,j,nr,nc);`
Operazione che disegna un rettangolo di blocchi sul livello `s`. Il rettangolo di blocchi ha come limite inferiore-sinistro la casella (*i*,*j*) ed è alto `nr` righe e largo `nc` colonne. La funzione può essere invocata solo a gioco fermo, e il rettangolo deve entrare completamente nel livello. Se tali condizioni non si verificano, la funzione non ha effetto. Deve essere possibile concatenare le chiamate a `blocchi`, per esempio: `s.blocchi(1,2,3,4).blocchi(5,6,7,8)`. La funzione elimina eventuali altri elementi (es. anelli) presenti in precedenza nel rettangolo.

✓ `s.anello(i,j);`

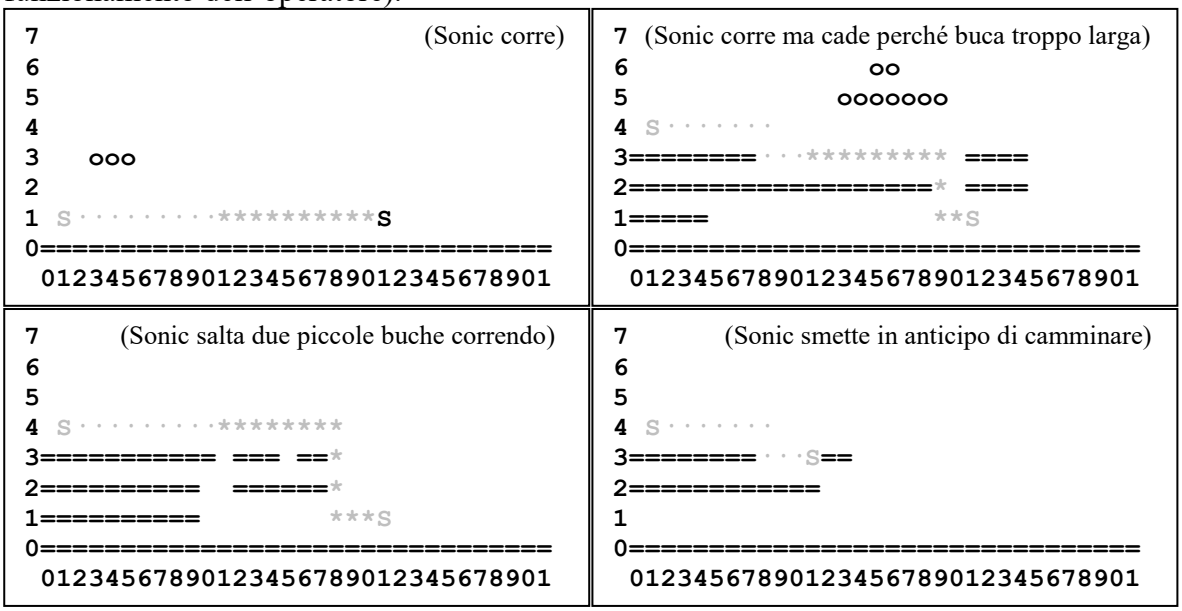
Operazione che piazza un anello in posizione (*i*,*j*) del livello `s`. L'operatore di uscita deve rappresentare la posizione dell'anello con un carattere '**o**' minuscolo. La funzione può essere invocata solo a gioco fermo e la posizione dell'anello deve essere una casella vuota. Se tali condizioni non si verificano, la funzione non ha effetto. Deve essere possibile concatenare le chiamate ad `anello`.

✓ `s.avvia(i,j);`

Operazione che avvia (o ri-avvia) il gioco sul livello `s`. Sonic viene collocato nella casella di partenza (*i*,*j*). Tale casella di partenza deve essere una casella vuota immediatamente sopra un blocco. Se tali condizioni non si verificano, la funzione non ha effetto. All'inizio del gioco, Sonic non ha raccolto nessun anello.

✓ `s += n;`

Operatore di somma e assegnamento tra un `SonicLevel` e un intero positivo, che fa camminare Sonic in avanti (cioè verso destra nel livello) di `n` caselle. Se durante la camminata Sonic 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. Se durante la camminata Sonic trova un blocco davanti a sé, egli smette in anticipo di camminare. Se Sonic cammina per più di 10 caselle senza smettere, allora inizia a correre. Mentre sta correndo, Sonic può passare sopra a piccole "buche", cioè se non trova più un blocco sotto ai piedi ma lo trova alla casella successiva Sonic continua a correre in orizzontale come se la buca non esistesse. Seguono quattro esempi di camminata, tutti di `n=20` caselle. La '**S**' grigia rappresenta la posizione iniziale di Sonic, quella nera la posizione finale. I caratteri '·' rappresentano caselle dove Sonic ha camminato, i caratteri '*****' dove ha corso (tali caratteri non devono essere stampati dall'operatore di uscita, servono solo per esemplificare il funzionamento dell'operatore).



Se Sonic passa attraverso un anello durante la camminata o la corsa o la caduta, egli lo raccoglie ed esso sparisce dal livello. L'anello non riappare nemmeno al riavvio del gioco: deve essere ri-piazzato esplicitamente con il metodo `anello` a gioco fermo. Deve essere possibile concatenare le chiamate all'operatore di somma e assegnamento nel modo seguente: `(s+=2)+=4`. L'operatore può essere invocato solo a gioco avviato, e l'intero `n` deve essere positivo. Se tali condizioni non si verificano, l'operatore non ha effetto.

--- SECONDA PARTE ---

✓ `~SonicLevel();`

Qualora sia necessario, implementare il distruttore.

✓ `s.spuntone(i,j);`

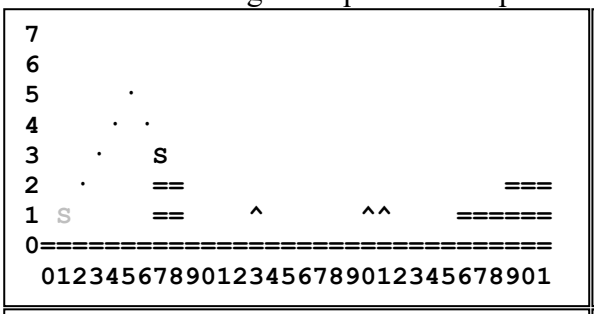
Operazione che piazza uno spuntone in posizione (*i,j*) nel livello `s`. L'operatore di uscita deve rappresentare la posizione dello spuntone con un carattere '**^**'. La funzione può essere invocata solo a gioco fermo, e la posizione dello spuntone deve essere una casella vuota od occupata da un blocco. Se tali condizioni non si verificano, la funzione non ha effetto. Non è necessario che lo spuntone abbia un blocco sotto: esso può stare sospeso nell'aria. Lo spuntone resta immobile durante il gioco, ed è impenetrabile da Sonic, come un blocco. Deve essere possibile concatenare le chiamate a `spuntone`.

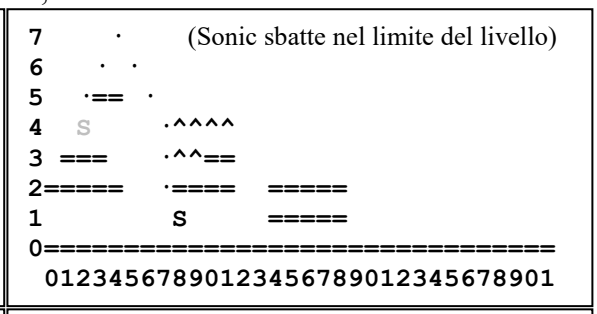
✓ `s += n;` (MODIFICA)

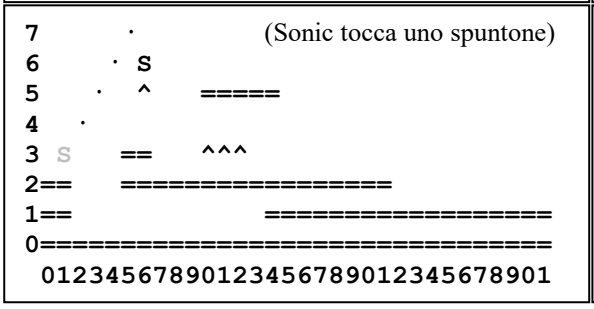
Modificare l'operatore di somma e assegnamento per tenere conto degli eventuali spuntoni presenti nel livello. Se Sonic tocca uno spuntone (cioè passa dalla casella superiore ad esso) ma ha raccolto degli anelli, perde tutti i suoi anelli ma sopravvive. In questo caso, per il resto del suo movimento è invulnerabile agli spuntoni e ci interagisce come normali blocchi. Se invece Sonic tocca lo spuntone e non ha anelli, muore ed il gioco si ferma.

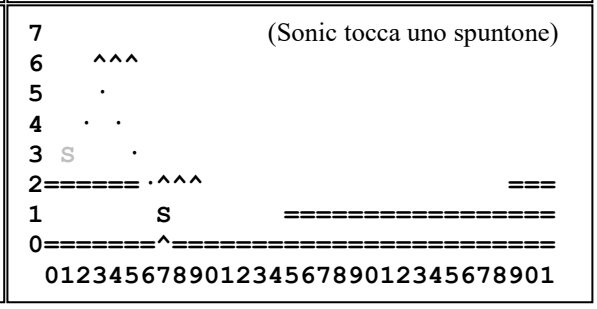
✓ `s *= n;`

Operatore di prodotto e assegnamento che fa saltare Sonic in avanti per un'altezza di `n` caselle, con `n` positivo. Il salto è composto da una fase ascendente in cui Sonic sale, cioè si sposta verso destra e in su, e una fase discendente in cui Sonic scende, cioè si sposta verso destra in giù. La fase ascendente dura per `n` caselle, o fino a quando Sonic non può più salire perché trova un blocco, uno spuntone, o la fine del livello. La fase discendente dura finché Sonic non trova un blocco sotto ai piedi. Se durante il salto il movimento di Sonic 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). Durante un salto Sonic può raccogliere anelli e toccare degli spuntoni, nel modo consueto. Seguono quattro esempi di salto, tutti di `n=4` caselle.









Deve essere possibile concatenare le chiamate all'operatore di prodotto e assegnamento. L'operatore può essere invocato solo a gioco avviato, e l'intero `n` deve essere positivo. Se tali condizioni non si verificano, l'operatore 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.

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---
Test del costruttore, di avvia e di operator<<
Anelli:0
7
6
5
4
3
2
1 S
0================================
 01234567890123456789012345678901
Test di operatore +=
(GIOCO FERMO)
Test di blocchi, anello, e avvia (ancora)
Anelli:0
7
6
5
4  S  o
3  =====
2  =====o
1  =====o o  ===
0================================
 01234567890123456789012345678901
Test di raccolta anelli, caduta e camminata fermata da muro
Test di riavvio
Anelli:0
7
6
5
4  S
3  =====
2  =====
1  =====     ===
0================================
 01234567890123456789012345678901
Test di uno schema con buche
Anelli:0
7
6
5
4 S
3========
2============== == =  ====
1=========================
0================================
 01234567890123456789012345678901
Anelli:0
7
6
5
4
3========
2============== == = S====
1=========================
0================================
 01234567890123456789012345678901
Test di uno schema con buche in cui Sonic non fa in tempo a correre
Anelli:0
7
6
5
4 S
3====== =
2============== == =  ====
1=========================
0================================
 01234567890123456789012345678901
Anelli:0
7
6
5
4
3======S=
2============== == =  ====
1=========================
0================================
 01234567890123456789012345678901
```

```
--- SECONDA PARTE ---
Test di operatore *=
Anelli:0
7
6              ====
5              ====
4
3====== =
2==============S== =  ====
1=========================
0=================================
 012345678901234567890123456789 01
Anelli:0
7
6              ====
5              ====
4
3====== =  S
2============== == =  ====
1=========================
0=================================
 012345678901234567890123456789 01
Anelli:0
7
6              ====
5              ====
4
3====== =                  S
2============== == =  ====
1=========================
0=================================
 012345678901234567890123456789 01
(GIOCO FERMO)
Test dello schema 1 con degli spuntoni
(GIOCO FERMO)
(GIOCO FERMO)
(GIOCO FERMO)
Test dello schema 1 con degli spuntoni e degli anelli
Anelli:0
7
6
5
4  So                   o
3  ==^==
2  =====    o            ^
1  =====     ===
0===============^==^==============
 012345678901234567890123456789 01
Anelli:0
7
6
5
4
3  ==^==                 S
2  =====                 ^
1  =====     ===
0===============^==^==============
 012345678901234567890123456789 01
(GIOCO FERMO)
```

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**.
