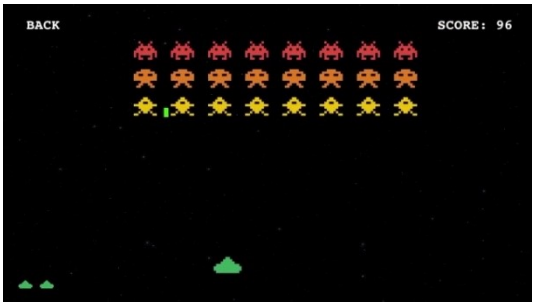

`SpaceAsteroids` è un videogioco ispirato al famoso gioco singolo giocatore "Space Invaders", con asteroidi al posto degli alieni. L'utente è chiamato a controllare un'astronave collocata alla base dello schermo (di dimensione al più 7 righe per 9 colonne) permettendole di viaggiare nello spazio evitando o distruggendo asteroidi lungo il cammino.

Implementare le seguenti operazioni effettuabili su uno `SpaceAsteroids`:

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

✓ **`SpaceAsteroids s(altezza, larghezza, energia_massima);`**

Costruttore che inizializza uno `SpaceAsteroids` avente schermo di altezza `altezza` (almeno pari a 3) e larghezza `larghezza` (quest'ultima sempre dispari ed almeno pari a 3). Eventuali valori scorretti vanno sanitizzati con il proprio valore massimo. L'inizializzazione prevede il collocamento dell'astronave in basso al centro dello schermo. Non vi sono asteroidi. L'implementazione deve prevedere anche l'inizializzazione del punteggio della partita a zero e dell'energia rimanente per il laser dell'astronave (si veda seconda parte) al suo valore massimo `energia_massima` (in caso di input scorretto, cinque).

NOTA: Lo studente capace di fornire una soluzione **che non fa uso di variabili globali** verrà premiato con **un punto di bonus.** Le costanti globali, invece, possono essere utilizzate, senza alcun problema.

✓ **`s.colloca_asteroide(col);`**

Metodo che colloca un asteroide nella riga più in alto dello schermo lungo la colonna `col` (indicizzata da 1). Il collocamento deve ritenersi possibile solo qualora lo spazio occupato sia vuoto. In caso di fallimento, il metodo restituisce `false`, altrimenti `true`.

✓ **`avanza_asteroidi();`**

Metodo *privato* che implementa l'avanzamento dell'astronave avanti nello spazio. Questo avanzamento è reso sullo schermo non con uno spostamento in avanti dell'astronave, bensì con lo spostamento degli asteroidi verso il basso di una riga (possono anche invadere l'ultima riga in questo caso). Qualora l'avanzamento degli asteroidi produca la collisione dell'astronave con uno di essi, il gioco si interrompe ed una nuova partita viene caricata coerentemente con quanto specificato per il costruttore (quindi senza asteroidi). Altrimenti, il punteggio dell'attuale partita viene incrementato di uno e, qualora superi l'attuale massimo punteggio ottenuto *in tutte le partite di tutti gli* `SpaceAsteroids`, incrementi anche tale record. Eventuali asteroidi che già si trovano nella riga più in basso dello schermo devono essere rimossi da esso in quanto ormai superati dall'astronave.

✓ **`s.avanza();`**

Metodo *pubblico* che per adesso invoca solamente la `avanza_asteroidi`. Il resto della sua implementazione è da realizzare nella seconda parte.

✓ **`cout << s;`**

Operatore di uscita per il tipo `SpaceAsteroids`. Segue un esempio di stampa di uno `SpaceAsteroids` di dimensione 7x7 prima (sinistra) e dopo (destra) l'invocazione del metodo `s.avanza()`:

```
Punteggio: 12
Record: 50
Energia: 5
_______
X  XXX
   XXX
 X XXX
   X


___A___
```

```
Punteggio: 13
Record: 50
Energia: 5
_______

X  XXX
   XXX
 X XXX
   X

___A___
```

Come si può notare, gli asteroidi sono indicati dalla lettera maiuscola "X", l'astronave dalla lettera maiuscola "A" e le parti superiori e inferiori dello schermo sono identificate da caratteri "_" (tranne dove si trova l'astronave o eventuali asteroidi, caso in cui viene visualizzato il corrispondente simbolo). Il punteggio corrente della partita è 12 (poi 13), il record tra tutte le partite di tutte le istanze di `SpaceAsteroids` è 50, l'energia rimanente all'astronave per lanciare raggi laser è 5.

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

✓ **`~SpaceAsteroids();`**

Implementare il distruttore se necessario.

✓ **`s <<= n; s >>= n;`**

Operatori di shift ed assegnamento per il tipo `SpaceAsteroids` che implementano lo spostamento laterale a sinistra (destra) dell'astronave di un numero di colonne pari ad `n` o fino a che non raggiunge il limite sinistro (destro) dello schermo, a seconda di cosa si verifichi prima. Due spostamenti laterali distinti (anche se nella stessa direzione) devono sempre essere separati da almeno una chiamata alla funzione `avanza` (si veda sotto). In caso contrario, tutte le invocazioni dopo la prima non hanno effetto. Qualora durante lo spostamento l'astronave collida con un asteroide, il gioco si interrompe e riavvia coerentemente con quanto descritto nel metodo `avanza_asteroidi`.

✓ **`s |= n;`**

Operatore di or bit a bit ed assegnamento per il tipo `SpaceAsteroids` che permette di sparare un raggio laser di intensità (positiva) `n` lungo la colonna in cui si trova attualmente l'astronave. Questo ha l'effetto di decrementare di altrettante unità l'energia rimanente all'astronave per i laser. Qualora `n` dovesse superare l'energia attualmente a disposizione (si veda il costruttore), utilizzarne quanta più possibile, sebbene in questo modo il raggio sarà meno intenso di quanto richiesto. Se l'astronave non ha energia, l'operatore non ha effetto.

Il rilascio di un raggio laser si manifesta a video con la comparsa del carattere "|" (pipe) immediatamente sopra l'astronave, ovvero nella stessa esatta colonna ma nella riga superiore. Il seguente esempio mostra uno `SpaceAsteroids` di dimensione 7x7 prima (sinistra) e dopo (destra) l'invocazione del metodo `s|=2`:

```
Punteggio: 12
Record: 50
Energia: 5

_______
X  XXX
   XXX
 X XXX
   X


___A___
```

```
Punteggio: 12
Record: 50
Energia: 3

_______
X  XXX
   XXX
X XXX
   X

   |
___A___
```

Gestire il caso in cui il raggio venga a sovrapporsi con un asteroide analogamente con quanto specificato nel metodo `avanza` a riguardo dell'impatto di un asteroide con un raggio laser. Come per gli spostamenti laterali, due invocazioni consecutive di questo operatore devono essere sempre intervallate da almeno una chiamata alla funzione `avanza`, altrimenti tutte le invocazioni successive alla prima devono essere ignorate. In caso di errore negli input, la struttura dati rimane inalterata.

✓ **`s.avanza();`**

Metodo che implementa l'avanzamento nello spazio dell'astronave e dei raggi laser. Quest'ultimi devono essere spostati verso l'alto di una riga lungo la colonna dove sono stati rilasciati. Per i raggi laser locati nella riga più alta dello schermo, l'avanzamento consiste semplicemente nella loro rimozione in quanto non più visibili. Contestualmente, il metodo provvede anche a far recuperare una unità di energia all'astronave. Dopo aver gestito la progressione dei raggi laser, il metodo invoca la funzione `avanza_asteroidi`, come già specificato nella prima parte.

Ogniqualvolta un raggio laser entri in contatto con un asteroide, quest'ultimo viene distrutto (ovvero rimosso dallo schermo), l'intensità del raggio diminuisce di uno ed il punteggio corrente della partita viene incrementato di uno. I raggi con intensità nulla si dissolvono (ovvero scompaiono dallo schermo). Qualora il punteggio corrente superi il record, provvedere ad aggiornare anche quest'ultimo.

Mediante il linguaggio C++, realizzare il tipo di dato astratto **`SpaceAsteroids`**, definito dalle precedenti specifiche. **Gestire le eventuali situazioni di errore.** Non è permesso utilizzare funzionalità della libreria STL come il tipo `string`, il tipo `vector`, il tipo `list`, ecc. Le librerie cstring, cmath e cstdlib, invece, possono essere utilizzate tranquillamente.

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---

Test del costruttore:
Punteggio: 0
Record: 0
Energia: 3
_______



___A___


Test della colloca_asteroide:
111

Punteggio: 0
Record: 0
Energia: 3
_______
X  X  X


___A___


Test della avanzamento_asteroidi:
Punteggio: 1
Record: 1
Energia: 3
_______

X  X  X


___A___
```

```
--- SECONDA PARTE ---

Test operatore <<=:
Punteggio: 1
Record: 1
Energia: 3
_______

X  X  X


A______


Test operatore |=:
Punteggio: 1
Record: 1
Energia: 1
_______

X  X  X


|
A______


Test della avanza:
Punteggio: 2
Record: 2
Energia: 2
_______

X  X  X
|

A______


Altro test della avanza:
Punteggio: 4
Record: 4
Energia: 3
_______


|
   X  X

A______
```

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**.