Un `Monitor` mostra una sequenza di messaggi, uno per riga, *dal più recente al più vecchio*. Ogni messaggio è costituito da al più `MAXLEN=10` caratteri. Il numero massimo di messaggi che un monitor può mostrare è costante. Quando un `Monitor` ha raggiunto il numero massimo di messaggi, il prossimo messaggio va a sovrascrivere il più vecchio. Implementare le seguenti operazioni che possono essere effettuate su un `Monitor`.

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

- ✓ **`Monitor m(N);`**

Costruttore che inizializza un `Monitor` capace di visualizzare un numero massimo di messaggi pari a `N` (*capienza*). Se `N` è minore o uguale a zero, viene creato un `Monitor` con capienza 3 messaggi.

- ✓ **`m.inserisci(msg);`**

Operazione che inserisce il messaggio `msg` nel `Monitor` `m`. Se necessario, il messaggio `msg` viene troncato a `MAXLEN` caratteri. È possibile utilizzare la funzione `strncpy(dest,src,len)` della libreria `cstring`, la quale copia al più `len` caratteri da `src` a `dest`.

- ✓ **`cout << m;`**

Operatore di uscita per il tipo `Monitor`. L'output è nel formato seguente:

```
[4]
rockerduck
nonnapaper
paperone
paperino
```

L'esempio mostra un `Monitor` con capienza quattro messaggi. La prima riga visualizza la capienza racchiusa tra parentesi quadre. Le successive righe visualizzano i messaggi presenti nel `Monitor`, dal più recente `("rockerduck")` al più vecchio `("paperino")`.

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

- ✓ **`Monitor m1(m);`**

Costruttore di copia, che crea un `Monitor m1` uguale a `m`.

- ✓ **`m1 + m2;`**

Operatore di somma che produce un `Monitor` la cui capienza è la somma delle capienze degli operandi, e i cui messaggi sono i messaggi del primo operando più quelli del secondo. I messaggi del secondo operando devono apparire come quelli *più recenti* nel `Monitor` risultato.

- ✓ **`~Monitor();`**

Distruttore.

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

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---
Test del costruttore:
[4]

Test di inserisci:
[4]
paperino
pluto
pippo

Altro test di inserisci:
[4]
rockerduck
nonnapaper
paperone
paperino

--- SECONDA PARTE ---
Test del costruttore di copia:
[4]
rockerduck
nonnapaper
paperone
paperino

Test di operator+:
[7]
gambadileg
topolino
rockerduck
nonnapaper
paperone
paperino

[7]
qua
quo
qui
gambadileg
topolino
rockerduck
nonnapaper

Test del distruttore:
(m3 e' stato distrutto)
```

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**.