Un `Display` a caratteri è costituito da `L` righe, ciascuna composta da `C` caratteri. Sul display è possibile scrivere una riga per volta, a partire dalla prima riga in alto. Dopo ogni scrittura, il cursore scorre alla riga successiva, e la prossima scrittura verrà fatta su quella riga. Quando viene scritta l'ultima riga in basso, il cursore ritorna alla prima riga in alto, e la prossima scrittura sovrascriverà quella riga.
Implementare le seguenti operazioni che possono essere effettuate su un `Display`:

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

✓ **`Display d(L,C);`**

Costruttore che inizializza un `Display` con `L` righe, ciascuna composta da `C` caratteri. Inizialmente, tutte le righe del display sono vuote. Se almeno uno dei valori di ingresso non è valido (il display deve avere almeno una riga e un carattere), si imponga `L=5` e `C=8`.

✓ **`Display d1(d);`**

Costruttore di copia che inizializza un nuovo `Display d1`, con il valore del `Display d`.

✓ **`d.writeT(str);`**

Scrive la stringa `str` sul display `d` con troncamento: se la stringa è più lunga della riga, la stringa viene troncata. Se la stringa è vuota o nulla, la scrittura non viene effettuata e il display rimane inalterato.

✓ **`cout << d;`**

Operatore di uscita per il tipo `Display`. L'uscita ha il seguente formato:

```
[1]Lazio
[2]Toscan
[3]Umbria
[4>
```

L'output mostrato corrisponde a un `Display` avente quattro righe composte da sei caratteri. Il carattere '>' indica il cursore, ovvero la riga che verrà occupata dalla prossima scrittura.

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

✓ **`d.writeW(str);`**

Scrive la stringa `str` sul display `d` con traboccamento: se la stringa è più lunga della riga, la stringa va ad occupare anche le righe successive. Se la stringa è vuota o nulla, la scrittura non viene effettuata e il display rimane inalterato.

✓ **`d1 = d;`**

Operatore di assegnamento per il tipo `Display`, che assegna il valore del `Display d` al `Display d1`.

✓ **`~Display();`**

Distruttore.

Mediante il linguaggio C++, realizzare il tipo di dato astratto **`Display`**, definito dalle precedenti specifiche. **Gestire le eventuali situazioni di errore**.

continua sul retro del foglio

## OUTPUT ATTESO DAL PROGRAMMA

```
--- PRIMA PARTE ---
Test del costruttore:
[1>
[2]
[3]
[4]

Test di writeT:
[1]Lazio
[2]Toscan
[3]Umbria
[4>

Altro test di writeT:
[1]Sicili
[2>Toscan
[3]Umbria
[4]Sardeg

Test del costruttore di copia:
[1]Sicili
[2>Toscan
[3]Umbria
[4]Sardeg

--- SECONDA PARTE ---
Test di writeW:
[1]Marche
[2>Emilia
[3]Romagn
[4]a

Test dell'op. di assegnamento:
[1]Marche
[2>Emilia
[3]Romagn
[4]a

Test del distruttore:
(d2 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**.