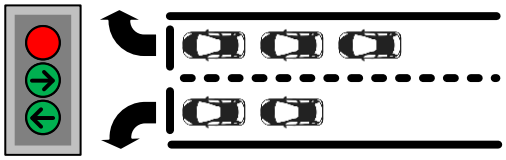

Un incrocio stradale è regolato da un `Semaforo`. Le auto che giungono all'incrocio si sistemano su *due corsie*: la prima è riservata alle auto che vogliono svoltare a destra, mentre la seconda è riservata alle auto che vogliono svoltare a sinistra. Un'auto viene univocamente identificata dalla propria targa, ovvero da una sequenza di 7 caratteri alfanumerici. In ogni istante, il semaforo può trovarsi in uno dei seguenti *tre stati*: rosso, verde per corsia destra, verde per corsia sinistra.
Implementare le seguenti operazioni che possono essere effettuate su un `Semaforo`:

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

✓ **`Semaforo s;`**

Costruttore che crea un `Semaforo`. Inizialmente, il semaforo è rosso e non ci sono auto in coda.

✓ **`s.arrivo(t, d);`**

Operazione che implementa l'arrivo al semaforo di un'auto con targa `t` e direzione `d`. La direzione può essere `'D'` o `'S'`, a seconda che l'auto si posizioni sulla corsia destra o sinistra. Se il semaforo è verde per tale corsia, l'auto attraversa immediatamente l'incrocio. In caso contrario, l'auto si mette in coda nella rispettiva corsia. Se `t` è già presente in una delle due corsie, lo stato delle code resta invariato.

✓ **`s.cambiaStato();`**

Operazione che cambia lo stato del semaforo, in base alla sequenza seguente:

- se il semaforo è rosso, diventa verde per la corsia destra;
- se il semaforo è verde per la corsia destra, diventa verde per la corsia sinistra;
- se il semaforo è verde per la corsia sinistra, diventa rosso.

Se il nuovo stato del semaforo è verde per una delle due corsie, tutte le eventuali auto in coda in quella corsia attraversano l'incrocio.

✓ **`cout << s;`**

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

```
<Rosso>
[AB123CD,CT789ZF
[BE456MH
```

Nell'esempio, il semaforo è rosso. Due auto sono in coda nella corsia destra e una è in coda nella corsia sinistra.

```
<Verde Destra>
[
[BE456MH
```

In quest'altro esempio, il `Semaforo` è verde per la corsia destra e c'è un'auto in coda nella corsia sinistra.

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

✓ **`s.cambiaCorsia(d);`**

Operazione che permette all'auto che si trova in ultima posizione nella coda della corsia relativa alla direzione `d` di spostarsi nell'altra corsia. Il cambio è permesso solo se la lunghezza della coda di destinazione è minore della lunghezza della coda di partenza. In tal caso, l'operazione restituisce `true`. In caso contrario, lo stato delle code rimane invariato e l'operazione restituisce `false`. Per esempio, si consideri la seguente situazione:

```
<Rosso>
[AB123CD,CT789ZF
[BE456MH
```

continua sul retro del foglio

La chiamata `s.cambiaCorsia('D')` ha come risultato:

```
<Rosso>
[AB123CD
[BE456MH,CT789ZF
```

Nel caso la nuova corsia permettesse l'attraversamento dell'incrocio (ovvero, il semaforo verde per la corsia di destinazione), l'auto attraversa immediatamente l'incrocio.

- ✓ **`int(s);`**

Operatore di conversione a intero per il tipo `Semaforo`, che restituisce il numero totale di auto in coda.

- ✓ **`~Semaforo();`**

Distruttore.

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

OUTPUT ATTESO DAL PROGRAMMA

```
--- PRIMA PARTE ---
Test del costruttore:
<Rosso>
[
[

Test della arrivo:
<Rosso>
[AB123CD,CT789ZF
[BE456MH

Test della cambiaStato:
<Verde Destra>
[
[BE456MH

Ulteriori test di arrivo e cambiaStato:
<Verde Sinistra>
[BL452MT
[

<Rosso>
[BL452MT,BV175TR
[AR271SD

--- SECONDA PARTE ---
Test della cambiaCorsia:
1
0
<Rosso>
[BL452MT
[AR271SD,BV175TR

Test di operator int:
Sono presenti 3 auto in coda

Test del distruttore:
<Rosso>
[CY379PN,FB478KG
[AV170MN

(s1 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 entro il termine comunicato in sede d'esame.
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**.