Un `AlberoDiNatale` è composto da `N` piani e ogni piano ha un numero di rami definito dalla seguente regola: il piano più in basso (di indice `0`) ha `N` rami, il piano superiore (di indice `1`) ha `N-1` rami, e così via fino al piano più in alto (di indice `N-1`), che ha un solo ramo. Per ogni piano, i rami sono a loro volta numerati a partire da `0`. Su ogni ramo può essere presente una pallina di colore rosso, verde o blu.
Implementare le seguenti operazioni che possono essere effettuate su un `AlberoDiNatale`:

--- PRIMA PARTE --- *(qualora siano presenti errori di compilazione, collegamento o esecuzione in questa parte, **l'intera prova sarà considerata insufficiente** e pertanto **non sarà corretta**)*

✓ **`AlberoDiNatale a(N);`**

Costruttore che crea un `AlberoDiNatale` avente `N` piani. Inizialmente, non ci sono palline sui rami dell'albero.

✓ **`a.aggiungiPallina(c, p, r);`**

Operazione che aggiunge ad `a` la pallina di colore `c` sul ramo `r` del piano `p`. L'operazione fallisce (ovvero, non modifica l'albero) se:

- il ramo indicato è già occupato da un'altra pallina, oppure
- una pallina dello stesso colore è già presente in un ramo adiacente dello stesso piano.

✓ **`cout << a;`**

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

```
   R
  - -
 - - V
- B V -
   |
```

L'output mostrato corrisponde a un `AlberoDiNatale` di 4 piani, avente una pallina blu sul ramo 1 del piano 0, una pallina verde sul ramo 2 del piano 0, una pallina verde sul ramo 2 del piano 1 e una pallina rossa sul ramo 0 del piano 3. Tutti gli altri rami (rappresentati dal carattere `'-'`) non hanno palline. L'operatore deve mostrare il carattere `'|'` in posizione centrale sotto l'ultima riga, al fine di rappresentare il tronco dell'albero. Tra due rami dello stesso piano è presente uno spazio.

--- SECONDA PARTE ---

✓ **`a += k;`**

Operatore di somma e assegnamento, che aggiunge `k` piani ad `a`. I nuovi piani vengono aggiunti in basso e inizialmente non hanno palline sui rami. Per esempio, effettuando l'operazione `a+=2` sull'albero dell'esempio precedente, l'albero che si ottiene è il seguente:

```
     R
    - -
   - - V
  - B V -
 - - - - -
- - - - - -
     |
```

✓ **`a.coloreMassimo();`**

Operazione che conta il numero di palline rosse, verdi e blu presenti sull'albero `a` e restituisce il carattere corrispondente al colore più rappresentato. In caso di parità, visualizzare il colore massimo secondo l'ordine di priorità decrescente `'R'`,`'V'`,`'B'`.

✓ **`~AlberoDiNatale();`**

Distruttore.

continua sul retro del foglio

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

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---
Test del costruttore:
   -
  - -
 - - -
- - - -
   |

Test della aggiungiPallina:
   R
  - -
 - - V
- B V -
   |

--- SECONDA PARTE ---
Test operator+=:
     R
    - -
   - - V
  - B V -
 - - - - -
- - - - - -
     |

     R
    - -
   R - V
  - B V -
 - - - R -
- - B - - -
     |

Test della coloreMassimo:
R

Test del distruttore:
(a1 e' stato distrutto)
```