Il tipo di dato `Museo` mantiene informazioni sulle opere d'arte esposte. Ogni opera d'arte è identificata da un codice univoco che consiste in una stringa di 5 caratteri, da un tipo di opera, dall'anno in cui e' stata prodotta, e dal numero di sala in cui è esposta. I possibili tipi delle opere sono: QUADRO, STATUA. Le sale sono numerate a partire da 1. Considerare le seguenti operazioni definite su `Museo`:

- ✓ **`Museo m;`**

Costruttore di default che inizializza un museo m che può mantenere al più 100 opere d'arte e che ha 10 sale. All'inizio il museo è vuoto.

- ✓ **`Museo m(n, k);`**

Costruttore che inizializza un museo `m` che può mantenere al più `n` opere d'arte e che ha `k` sale. All'inizio il museo è vuoto.

- ✓ **`Museo m1(m);`**

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

- ✓ **`m.aggiungi_opera(id, t, a, j);`**

Operazione che aggiunge un'opera d'arte con codice `id`, di tipo `t`, prodotta nell'anno `a` nella sala `j` del museo `m`. L'operazione restituisce `true` se l'inserimento ha successo, `false` altrimenti.

- ✓ **`m.elimina_opera(id);`**

Operazione che elimina l'opera il cui codice è uguale a `id` dal museo `m`. L'operazione restituisce `true` se l'eliminazione ha avuto successo, `false` altrimenti.

- ✓ **`m.sposta(id,i,j);`**

Operazione che sposta l'opera il cui codice è uguale a `id` dalla sala `i` alla sala `j`.

- ✓ **`m%i;`**

Operatore modulo che restituisce il numero di opere esposte nella sala `i` del museo `m`.

- ✓ **`cout<<m;`**

Operatore di uscita per il museo. L'uscita visualizza, su righe diverse, il numero totale di opere, il numero delle sale e, per ogni tipo di opera, i codici delle opere di quel tipo separati dal carattere ','.
Ad esempio, l'uscita seguente corrisponde ad un museo `m` che ha 7 sale e che espone 3 opere di tipo QUADRO e nessuna opera di tipo STATUA.

Numero di sale: 7
Numero di opere: 3
QUADRO: ABBCC, BBBCA, AAABC
STATUA:

- ✓ **`~Museo();`**

Distruttore.

Mediante il linguaggio C++, realizzare il tipo di dato astratto `Museo`, definito dalle precedenti specifiche. Individuare eventuali situazioni di errore, e metterne in opera un corretto trattamento.

Di seguito viene riportato un esempio di main.cpp e la relativa uscita attesa.

```
// FILE main.cpp
#include "compito.h"
int main() {
      //test costruttore / distruttore
      {
            Museo m;
      }

      //test operator<<
      Museo m(5,3); //al masimo 5 opere d'arte in 3 sale
      cout << m << endl;

      //test aggiungi_opera
      m.aggiungi_opera("AAAAA", QUADRO, 1990, 3);
      m.aggiungi_opera("AAABB", QUADRO, 2003, 3);
      m.aggiungi_opera("AAABB", STATUA, 2003, 3);  // non viene inserito,
                                                   // codice gia' presente
      m.aggiungi_opera("AAACC", STATUA, 2003, 2);
      m.aggiungi_opera("AAADD", QUADRO, 2000, 2);
      cout << m << endl;

      //test elimina_opera
      m.elimina_opera("AAADD");
      cout << m << endl;

      //test sposta
      m.sposta("AAACC", 2, 1);
      cout << m << endl;

      //test operator%
      cout << m%3 << endl;

      return 0;
}
```

Uscita attesa

```
Numero di sale: 3
Numero di opere: 0
QUADRO:
STATUA:

Numero di sale: 3
Numero di opere: 4
QUADRO: AAAAA, AAABB, AAADD
STATUA: AAACC

Numero di sale: 3
Numero di opere: 3
QUADRO: AAAAA, AAABB
STATUA: AAACC

Numero di sale: 3
Numero di opere: 3
QUADRO: AAAAA, AAABB
STATUA: AAACC

0
```