Fondamenti di Programmazione 2
(Corso di Laurea in Informatica)
Esame del 13/7/2023

UNIVERSITÀ DELLA CALABRIA

Esercizio 1

Data la seguente porzione di programma rispondere alle domande corrispondenti:

```
#include <iostream>
using namespace std;

int main() {
    int *m = new int[6]{.. la tua matricola ..};

   // 1. Cosa stampa questa istruzione?
   cout << *(m + 3)-2 <<endl;
  // 2. La seguente istruzione è corretta? Se si, cosa stampa?
    m[0]=m[1]==9;
   cout << m[0] + m[1] + m[2] << endl;
// 3. Che valore viene stampato per la variabile somma?
   int somma=0;
   for (int i = 0;i<6;i++){
      if(i%2==0){
         int &a = m[i];
         a*=0;
      }
      somma += m[i];
   }
   cout<<somma;

// 4.La seguente istruzione è corretta? Se si, cosa stampa?Se no, perchè?
   delete[] m;
   cout << m[0] << endl;
}
```

Esercizio 2

Si consideri la seguente classe e completare:

```
class Spedizione{
      private:
          int codice;
          float valore;
          float peso;

      public:
          int getCodice() const;
          float getValore() const;
          float getPeso() const;

          // da implementare
          friend ostream& operator<<(ostream& os,
          const Spedizione& sp);
          bool operator==(const Spedizione&);
};
```

Implementare **operator<<** e **operator==** per lla classe Spedizione.

Implementare quindi la classe **CodaSpedizioni** che eredita opportunamente da list<Spedizione*>. Definire almeno i seguenti metodi:

void add(Spedizione*); *//aggiunge la spedizione se non esiste già, quindi se codice, valore e peso sono tutti differenti. La funzione deve garantire che le spedizioni nella coda siano ordinate: si consideri prima il peso, a parità di peso il valore, a parità di valore il codice.*

Spedizione* next() const;
//restituisce il prossimo elemento della coda, senza rimuoverlo

void remove();
//rimuove il prossimo elemento nella coda

unsigned int size() const.

//restituisce il numero di elementi nella coda

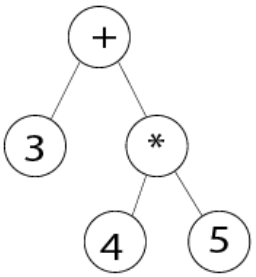
UNIVERSITÀ DELLA CALABRIA

Esercizio 3

Ogni espressione aritmetica può essere codificata da (almeno) un albero binario dove le foglie hanno come valore informativo numero intero e i nodi interni un operatore aritmetico tra "+" (addizione), "-" (sottrazione), "/" (divisione intera), e "*" (moltiplicazione).

Scrivere una funzione che presa in input un'espressione aritmetica, modellata da un'istanza di `AlberoB<std::string>`, restituisca il suo risultato (come `int`).

Si può assumere l'input sia valido (es. espressioni sintatticamente corrette, no divisioni per zero, …), ed è possibile utilizzare la funzione `int std::strtoi(std::string)` per convertire un'istanza di `std::string` nel corrispondente intero.

L'albero in figura codifica l'espressione aritmetica `3+(4*5)`. La funzione dovrà restituire 23. NB: Nell'albero in input, i valori informativi dei nodi sono rispettivamente le stringhe "+", "3", "*", "4", "5".	

NOTA: La classe template `AlberoB<T>` possiede la seguente interfaccia pubblica, dove `t` è un'istanza della classe:

- `t.radice()` - Restituisce il valore informativo della radice di `t`.
- `t.nullo()` - Restituisce `true` se `t` è l'albero vuoto, `false` altrimenti.
- `t.figlio(d)` - Restituisce il sottoalbero sinistro (se `d == SIN`) o destro (se `d == DES`).

Esercizio 4

Scrivere una funzione che presi in input un insieme finito di numeri interi S, degli insiemi finiti $\{C_1, C_2, \ldots, C_n\}$, con $C_i \subset S$, ed un intero $0 < k < n$, e restituisca true se e solo se è possibile generare un insieme S^* con almeno k elementi tali che ciascun $x \in S^*$ compaia in almeno k tra gli insiemi $\{C_1, C_2, \ldots, C_n\}$.

- Si può assumere che gli elementi di S e degli insiemi $\{C_1, C_2, \ldots, C_n\}$ siano tutti distinti;
- Si può assumere che S sia rappresentato come istanza di `std::vector<int>`;
- Si può assumere che C sia rappresentato come istanza di `std::vector<std::vector<int>>`.