Corso di Laboratorio di Programmazione

Prova intermedia di programmazione

1. Progettare e implementare la classe `Book` che rappresenta un libro. Tale classe ha membri per contenere:
   a. ISBN;
   b. titolo;
   c. nome e cognome autore (due membri diversi) – è previsto un solo autore per libro;
   d. data di copyright – la data è rappresentata usando una apposita classe `Date` simile a quella sviluppata a lezione;
   e. stato in prestito / disponibile.

Il codice ISBN è rappresentato mediante una `std::string` il cui unico requisito è quello di avere lunghezza 13.

La classe `Book` deve essere dotata di:

a. funzioni membro che ritornano i vari dati membro;
b. costruttori;
c. funzioni per registrare il prestito e la restituzione;
d. appositi strumenti (eventualmente implementati nella classe dedicata) per la validazione della data;
e. `operator==` e `operator!=` che confrontano due `Book` basandosi sul codice ISBN;
f. `operator<<` che stampa titolo, autore, ISBN e la data di copyright su righe separate in output.

La classe `Book` deve permette di eseguire la seguente operazione:

a. `Book mybook("David", "Foster Wallace", "Una cosa divertente che non farò mai più", "887-521-837-4");`

2. Modificare opportunamente la classe `MyVector` implementata durante i precedenti laboratori per creare una classe `BookShelf` che gestisce vettori di elementi `Book`. Tramite la classe `BookShelf` dovranno essere compilabili le seguenti istruzioni:

```
a. BookShelf shelf(10);      // Crea un vettore di 10 elementi Book
b. shelf.push_back(mybook); // Aggiunge l'elemento mybook al
                            // vettore shelf; mybook è un oggetto
                            // della classe Book precedentemente
                            // creato
c. shelf.pop_back();        // Rimuove l'ultimo elemento (se
                            // esiste) dal vettore shelf
```

La classe `Book`, la classe `BookShelf` (e le altre eventuali classi) devono essere correttamente separate nei file `.h` e `.cpp`. Un ulteriore file `.cpp` deve contenere il `main`, usato per i test. Il progetto deve essere opportunamente strutturato in cartelle e sottocartelle analogamente a quanto illustrato nello schema seguente (relativo al progetto `Rational`):

```
Rational:
├── CMakeLists.txt
├── include
│   └── rational.h
├── README.txt
└── src
    ├── main.cpp
    └── rational.cpp
```

Vista la complessità del progetto è fortemente consigliato l’utilizzo di CMake.