FONDAMENTI DI PROGRAMMAZIONE B

Tempo a disposizione: 2 ore 30 minuti

Nome ..... Cognome ..... Matricola .....

Esercizio 1 [C++] (15pt). Definire una classe templatica `Queue<T>` che realizza il tipo di dato astratto coda di elementi di tipo `T` che permette di inserire un nuovo elemento in coda e di rimuovere un elemento dalla testa della coda.

La classe deve definire:

- un costruttore senza argomenti che crea una coda vuota
- un metodo `enqueue` che, preso come argomento `elem`, aggiunge `elem` in coda
- un metodo `dequeue` che rimuove l'elemento in testa alla coda e lo restituisce. Se la coda è vuota, il metodo deve lanciare un'eccezione.
- un metodo `isEmpty` che controlla se la coda è vuota

Si sovraccarichi l'operatore `<<` in modo tale che stampi gli elementi della coda su uno stream di output `fout` nel formato $[e_1, e_2, \ldots, e_n]$. Non è consentito utilizzare classi della STL. Se necessario, ridefinire gli opportuni metodi, costruttori e/o operatori. Specificare opportunamente eventuali metodi e parametri costanti. Massimizzare incapsulamento e information hiding.

Esercizio 2 [Java] (15pt). Definire una classe generica `Stack<T>` che realizza il tipo di dato astratto pila di elementi di tipo `T` (LIFO: Last In First Out). La classe deve definire:

- un costruttore senza argomenti che crea una pila vuota
- un metodo `push` che aggiunge un elemento di tipo `T` alla pila
- un metodo `pop` che rimuove dalla pila l'ultimo elemento inserito e lo ritorna come risultato. Il metodo deve lanciare un'eccezione di tipo **controllato** `EmptyStackException` (da implementare) se la pila è vuota
- un metodo `peek` che ritorna l'elemento in cima alla pila senza rimuoverlo. Il metodo deve lanciare l'eccezione `EmptyStackException` se la pila è vuota
- un metodo `isEmpty` che controlla se la pila è vuota
- un metodo `size` che ritorna il numero di elementi memorizzati nella pila

La classe deve sovrascrivere il metodo `toString` e deve ritornare una stringa rappresentante la pila nel formato $[e_1, e_2, ..., e_n]$. La classe deve sovrascrivere il metodo `equals`. Massimizzare incapsulamento e information hiding.
(+2pt) La classe `Stack<T>` deve implementare l'interfaccia `Comparable`. Il metodo corrispondente da implementare utilizza il numero di elementi nella pila per il confronto.