FONDAMENTI DI PROGRAMMAZIONE B

Tempo a disposizione: 2 ore 30 minuti

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

Esercizio 1 [C++] (15pt). Definire una classe templatica `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. La classe deve definire un metodo `push` che aggiunge un elemento di tipo `T` alla pila. La classe deve definire un metodo `pop` che rimuove dalla pila l'ultimo elemento inserito e lo ritorna come risultato. Il metodo deve lanciare un'eccezione se la pila è vuota. La classe deve definire un metodo `isEmpty` che controlla se la pila è vuota. La classe deve definire un metodo `size` che ritorna il numero di elementi memorizzati nella pila. Si sovraccarichi l'operatore `<<` in modo tale che stampi gli elementi della pila su uno stream di output `fout` nel formato $[e_1, e_1, \ldots, e_n]$. 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). Nel contesto dello sviluppo di un social network si sviluppino le seguenti classi e interfacce.

- Si implementi un'interfaccia `User`, che contiene solamente un metodo senza parametri `getUsername` che ritorna un oggetto di tipo `String`. Si implementino le classi `NormalUser`, che modella un utente normale, e `PremiumUser`, che modella un utente di tipo premium. Entrambe le classi devono implementare l'interfaccia `User` descritta nel punto precedente.

- La classe `NormalUser` modella un utente di tipo normale. Un utente normale è caratterizzato da un nome, un cognome e un insieme di amici. La classe deve mettere a disposizione un unico costruttore che prende come argomenti il nome e il cognome dell'utente e inizializza i campi corrispondenti. Un oggetto di tipo `NormalUser`, quando costruito, non ha amici.

  Si implementi il metodo `addFriend` che prende come parametro un oggetto di tipo `User` e lo aggiunge all'insieme degli amici di `this`. Se l'oggetto passato come parametro è uguale a `this`, il metodo deve lanciare **un'eccezione di tipo non controllato** `SocialNetworkException`, da implementare.

  Si implementi il metodo `follow` che prende come parametro un oggetto di tipo `PremiumUser`. Tale metodo deve aggiungere `this` ai follower dell'utente premium passato come parametro. Se `this` è già un follower dell'utente premium, il metodo deve lanciare l'eccezione `SocialNetworkException`, definita precedentemente.

  Si ridefiniscano i metodi `equals` e `toString`. Due utenti normali sono uguali per il metodo `equals` se hanno lo stesso nome e lo stesso cognome. Il metodo `toString` deve stampare lo username di `this`. Lo username di un utente normale è la concatenazione del proprio nome e cognome seguito dal proprio numero di amici. È vietato utilizzare un campo apposito per lo username nella classe `NormalUser`.

- La classe `PremiumUser` modella un utente premium. Un utente premium è caratterizzato da uno username e un insieme di follower. La classe deve mettere a disposizione un unico costruttore che prende come argomento lo username dell'utente premium e inizializza il campo corrispondente. Un oggetto di tipo `PremiumUser`, quando costruito, non ha follower.

  Si implementi il metodo `addFollower` che prende un come parametro un oggetto di tipo `User` e lo aggiunge ai follower di `this`. Se l'utente passato come argomento è già un follower di `this`, il metodo deve lanciare l'eccezione `SocialNetworkException`, definita precedentemente.

  Si ridefiniscano i metodi `equals` e `toString`. Due utenti premium sono uguali per il metodo `equals` se hanno lo stesso username. Il metodo `toString` deve stampare lo username di `this`.

- Si implementi la classe `SocialNetwork` caratterizzata dal nome del social network e dagli utenti iscritti al social network. La classe deve mettere a disposizione un unico costruttore che prende come argomento il nome del social network. Un oggetto di tipo `SocialNetwork`, quando costruito, non ha utenti iscritti.

  Si implementi il metodo `addUser` che prende come parametro un oggetto di tipo `User` e lo aggiunge all'insieme di iscritti di di `this`. Se l'utente è già iscritto, il metodo deve lanciare l'eccezione `SocialNetworkException`, definita precedentemente.

N.B. Massimizzare incapsulamento e information hiding. Non è richiesta l'implementazione del metodo `hashCode` per le classi richieste. Per ciascuna classe, è possibile supporre di avere a disposizione un'implementazione del metodo `hashCode` coerente col metodo `equals` che implementerete.

(+2pt) La classe `PremiumUser` deve implementare l'interfaccia `Comparable<T>`. Il metodo corrispondente da implementare utilizza **il numero di follower** degli utenti premium coinvolti per il confronto.