FONDAMENTI DI PROGRAMMAZIONE B

Tempo a disposizione: 2 ore 15 minuti

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

Esercizio 1 [C++] (15pt). Definire una classe templatica `MultiSet<T>` che realizza il tipo di dato astratto multi-insieme di elementi di tipo `T` (informalmente, un multi-insieme è un insieme che ammette elementi ripetuti). La classe deve definire:

- un costruttore senza argomenti che crea un multi-insieme vuoto
- un metodo `add` che aggiunge un elemento preso come argomento al multi-insieme
- un metodo `remove` che, preso come argomento `elem`, rimuove dal multi-insieme tutti gli elementi uguali ad `elem`. Se il multi-insieme è vuoto, il metodo deve lanciare un'eccezione
- un metodo `isEmpty` che controlla se il multi-insieme è vuoto
- un metodo `cardinality` che, preso come argomento `elem`, ritorna il numero di occorrenze di `elem` nel multi-insieme

Inoltre, si sovraccarichi l'operatore `<<` in modo tale che stampi gli elementi del multi-insieme 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). Nel contesto di un registro delle iscrizioni in palestra, si sviluppino le seguenti classi.

- Si implementi una classe `Utente` che rappresenta un utente della palestra. Un utente è caratterizzato da un nome e da un cognome. 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. Si ridefiniscano i metodi `equals` e `toString`. Due utenti sono uguali per il metodo `equals` se hanno lo stesso nome e lo stesso cognome. Il metodo `toString` deve stampare il nome e il cognome di `this`.

- Si implementi una classe `Iscrizione` che rappresenta un'iscrizione in palestra. La classe è caratterizzata dall'utente che si vuole iscrivere, un mese e un anno di inizio dell'iscrizione (primo giorno del mese), un mese e un anno di fine dell'iscrizione (ultimo giorno del mese). La classe deve mettere a disposizione un unico costruttore che prende come argomenti un utente che si iscrive, il mese e l'anno di inizio dell'iscrizione, e il mese e l'anno di fine dell'iscrizione ed inizializza i campi corrispondenti. Se il mese e l'anno di inizio dell'iscrizione vengono cronologicamente dopo il mese e l'anno di fine **oppure** se l'iscrizione va oltre 12 mesi, viene lanciata un'eccezione di tipo non controllato `IscrizioneNonValidaException`, da implementare. La classe deve implementare un metodo `overlapsWith` che prende come parametro un'iscrizione `other` e determina se `this` e `other` sono sovrapposte, anche parzialmente, nel tempo.

  Si ridefiniscano i metodi `equals` e `toString`. Due iscrizioni sono uguali per il metodo `equals` se si riferiscono allo stesso utente e hanno stesso mese e anno di inizio e di fine.

- Si implementi la classe `Registro` che tiene traccia delle iscrizioni in una palestra. La classe deve mettere a disposizione un unico costruttore senza parametri. Un oggetto di tipo `Registro`, quando costruito, non ha iscrizioni.

  Si implementi il metodo `add` che prende come parametro un'iscrizione e la aggiunge al registro. Se il registro contiene già un'iscrizione **sovrapposta** per lo stesso utente, il metodo lancia l'eccezione `IscrizioneNonValidaException`, descritta precedentemente. Si implementi un metodo `project` che ritorna un nuovo registro che contiene tutte e sole le iscrizioni che si riferiscono ad un anno passato come parametro. Il metodo non deve modificare il registro su cui è invocato il metodo.

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 `Iscrizione` deve implementare l'interfaccia `Comparable<T>`. Un'iscrizione è minore di un'altra se il mese e l'anno di fine vengono cronologicamente prima.