FONDAMENTI DI PROGRAMMAZIONE B

Tempo a disposizione: 2 ore 30 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 `contains` che, preso come argomento `elem`, controlla se `elem` è contenuto nel multi-insieme
- un metodo `isEmpty` che controlla se il multi-insieme è vuoto

Inoltre, si sovraccarichi l'operatore `==` in modo tale che controlli se il multi-insieme su cui è chiamato il metodo è uguale ad un multi-insieme passato come parametro. Due multi-insiemi sono uguali se hanno gli stessi elementi e appaiono lo stesso numero di volte. 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 delle elezioni politiche, si sviluppino le seguenti classi ed interfacce.

- Si implementi un'interfaccia `Eleggibile` che contiene un metodo `getNome` che ritorna un oggetto di tipo `String` e un metodo `getVoti` che ritorna un intero. Si implementino le classi `Partito` e `Coalizione`. Entrambe le classi devono implementare l'interfaccia `Eleggibile`.
- La classe `Partito` è caratterizzata da un nome e dal numero di voti. La classe deve mettere a disposizione un costruttore che prende come parametro il nome del partito e lo inizializza. Quando construito, un partito non ha alcun voto. La classe deve mettere a disposizione un metodo `vota` che aggiunge un voto al partito. Il metodo `getVoti` ritorna il numero di voti di quel partito.

  Si ridefiniscano i metodi `equals` e `toString`. Due partiti sono uguali per il metodo `equals` se hanno lo stesso nome e lo stesso numero di voti.
- Si implementi la classe `Coalizione` che rappresenta una coalizione di uno o più partiti. La classe deve mettere a disposizione un costruttore che prende come parametro il nome della coalizione e un array di partiti di cui fa parte la coalizione ed inizializza i campi corrispondenti. Il metodo `getVoti` ritorna il numero di voti di quella coalizione, ottenuto sommando i voti di ciascun partito della coalizione.

  Si ridefiniscano i metodi `equals` e `toString`. Due coalizioni sono uguali per il metodo `equals` se hanno lo stesso nome e sono composte dagli stessi partiti.
- Si implementi la classe `Elezione` caratterizzata dai partiti e dalle coalizioni che vogliono partecipare all'elezione. La classe deve mettere a disposizione un costruttore senza parametri. Quando costruita, una elezione non ha nessun partito o coalizione che partecipa. La classe deve mettere a disposizione un metodo `add` che aggiunge come partecipante all'elezione un partito o una coalizione. Il metodo deve lanciare un'eccezione di tipo non controllato quando:
  - il partito o la coalizione è già iscritto all'elezione
  - il partito fa parte di una coalizione che è già iscritta all'elezione
  - la coalizione contiene un partito che è già iscritto all'elezione

  Si implementi il metodo `winner` che ritorna il partito o la coalizione che ha il maggior numero di voti.

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 una implementazione del metodo `hashCode` coerente col metodo `equals` che implementerete.

(+2pt) La classe `Partito` deve implementare l'interfaccia `Cloneable` e ridefinire il metodo `clone`.