Esame di Programmazione II

Appello di giorno 2 Marzo 2015
Università degli Studi di Catania - Corso di Laurea in Informartica

Testo della Prova

Definizione Iniziale.
Una *Lista con Molteplicità (MultiList)* è una lista in cui ogni nodo x ha un campo speciale che indica la molteplicità della chiave $key(x)$. Tale campo è indicato con il simbolo $mul(x)$. Esso rappresenta il numero di volte in cui il valore $key(x)$ è presente nell'insieme degli elementi inseriti nella lista. Se $mul(x) = t$ allora sono presenti t copie dell'elemento $key(x)$ nella lista.

Specifiche.
La corretta implementazione di ciascuno dei seguenti punti permette l'acquisizione di 9 punti. La corretta implementazione della classe come template è facoltativa e permette l'acquisizione di ulteriori 3 punti.

1. Si fornisca una classe `C++`, denominata `MyMultiList<H>`, che implementi la seguente interfaccia `MultiList<H>`, che rappresenta una lista con molteplicità ordinata in senso non decrescente, e contenente i seguenti metodi virtuali.

   (a) `MultiList<H>* ins(H x)` aggiunge un nuovo elemento (nella corretta posizione) alla struttura dati e restituisce un puntatore ad un oggetto di tipo `MultiList<H>`.
   (b) `int multiplicity(H x)` restituisce la molteplicità dell'elemento x, se questo è presente nella lista, 0 altrimenti;
   (c) `void print()` è una procedura che stampa in output gli elementi della lista, tenendo conto delle molteplicità

   Si crei quindi un'istanza di `MyMultiList<int>` e si inseriscano al suo interno i seguenti elementi:
   `10, 5, 8, 3, 5, 1, 6, 5, 8, 1, 12, 7`

   Si esegua in seguito la stampa dei valori inseriti nell'albero attraverso la procedura print. L'output del programma sarà quindi:
   `1, 1, 3, 5, 5, 5, 6, 7, 8, 8, 10, 12`

```
template <class H> class MyMultiList {
  public:
     virtual MultiList<H>* ins(H x) = 0;
     virtual int multiplicity(H x) = 0;
     virtual void print() = 0;
 }
```

2. Si inserisca all'interno della classe `MultiList<H>` l'implementazione della seguente procedura `MultiList<H>* del(H x)` che cancella un'occorrenza dell'elemento x dalla lista, se presente, e restituisce un puntatore ad un oggetto di tipo `MyMultiList<H>`. Si esegua in seguito la cancellazione, dall'istanza della struttura dati creata al passo precedente, degli elementi (nell'ordine) `6, 5, 8, 3, 8`. Si stampi in seguito il contenuto della lista. L'output del programma sarà quindi:

   `1, 1, 5, 5, 7, 10, 12`

3. Si aggiunga alla classe `MyMultiList<H>` un nuovo metodo denominato `rank`, la cui definizione è quella data di seguito. `int rank(H x)` Tale metodo prende in input un elemento di tipo `H` e restituisce in output il suo rango, ossia la sua posizione all'interno dell'insieme ordinato contenuto all'interno della struttura dati. Tale valore corrisponde al numero di elementi più piccoli di x, aumentato di uno. Il metodo restituisce il valore 0 se il valore non è presente nella struttura dati. Si esegua tre volte la procedura `rank` con input `5, 6` e `12`, rispettivamente. L'output del programma sarà quindi:

   `3, 0, 7`