Esame di Programmazione II

Appello di giorno 27 Giugno 2016
Università degli Studi di Catania - Corso di Laurea in Informatica
- PROVA A -

Testo della Prova

Definizione Iniaziale.
Una *Brik-List* è una struttura dati costituita da una lista i cui elementi sono delle code di valori numerici. Sia c una coda di valori numerici. Indichiamo con $\alpha(c)$ la somma degli elementi in essa contenuti, con $\beta(c)$ il numero di elementi in essa contenuti e con $\gamma(c)$ il valore che si trova in testa alla coda. Le code contenute all'interno della Brik-List sono ordinate, in ordine non decrescente, in base ai seguenti criteri:

- se $\alpha(c_1) > \alpha(c_2)$ allora $c_1 > c_2$
- se $\alpha(c_1) = \alpha(c_2)$ e $\beta(c_1) > \beta(c_2)$ allora $c_1 > c_2$
- se $\alpha(c_1) = \alpha(c_2)$, $\beta(c_1) = \beta(c_2)$ e $\gamma(c_1) > \gamma(c_2)$ allora $c_1 > c_2$

Un nuovo elemento può essere inserito all'interno della lista in due modi, in una nuova coda inizialmente vuota, oppure nella coda più piccola della Brik-List. Nella struttura dati non sono presenti code vuote. Quando da una coda viene estratto l'ultimo elemento, questa viene eliminata dalla lista. L'unico elemento che può essere eliminato dalla coda è l'elemento che si trova in testa alla coda più piccola della Brik-List.

Si fornisca una classe C++, denominata `MyBrikList<H>`, che implementi la seguente interfaccia `BrikList<H>`, implementata attraverso una lista doppiamente linkata e ordinata, contenente i seguenti metodi virtuali.

1. `BrikList<H>* ins(H x)` aggiunge una nuova coda alla BrikList, contenente il solo elmento x. Il numero di code della lista aumenta di uno. La funzione restituisce un puntatore ad un oggetto di tipo BrikList<H>;
2. `BrikListt<H>* push(H x)` aggiunge un nuovo elemento x alla prima coda della lista, cioè quella più piccola tra quelle presenti nella Brik-List. Il numero di code della lista non aumenta. La funzione restituisce un puntatore ad un oggetto di tipo BrikList¡H¿;
3. `BrikListt<H>* pop()` estrae il primo elemento dalla prima coda della lista, cioè quella più piccola tra quelle presenti nella Brik-List. Se l'elemento estratto è l'unico presente nella coda allora quest'ultima vien eliminata dalla lista. La funzione restituisce un puntatore ad un oggetto di tipo BrikList<H>;
4. `int search(H x)` restituisce 1 se x è presente nella struttura dati, 0 altrimenti;
5. `void print()` è una procedura che stampa in output gli elementi della lista. La stampa procede dalla prima all'ultima coda della lista. Per ogni coda gli elementi vengono stampati a partire dall'elemento in testa.

Nota Bene: la lista dovrà essere implementata attraverso una lista doppiamente linkata ed ordinata. La coda dovrà essere implementata come classe derivata dalla classe lista realizzata in precedenza.

Si crei quindi un'istanza di `MyBrikList<int>` e si inseriscano 10 nuove code contenenti rispettivamente i valori

15 12 6 9 10 4 2 30 23 11

Si esegua, in seguito, per quattro volte consecutive: l'estrazione di un elemento attraverso la procedura `pop()` e il suo re-inserimento nella struttura dati, attraverso la procedura `push()`.
L'output del programma sarà quindi:

```
4 2 6
15
10 9
23
12 11
30
```

• • •

```
template <class H> class BrikList {
  public:
     virtual BrikList<H>* ins(H x) = 0;
     virtual BrikList<H>* push(H x) = 0;
     virtual BrikList<H>* pop() = 0;
     virtual int search(H x) = 0;
     virtual void print() = 0;
 }
```

• • •

Valutazione.
La corretta implementazione della Lista permette di acquisire 9 punti. La corretta implementazione della coda permette di acquisire 9 punti. La corretta implementazione della class `MyBrikList` permette di acquisire ulteriori 9 punti. La corretta implementazione delle classi come template è facoltativa e permette l'acquisizione di ulteriori 3 punti:

• • •