Esame di Programmazione II

Appello di giorno 5 Ottobre 2016
Università degli Studi di Catania - Corso di Laurea in Informatica
- Prova B -

Testo della Prova

Definizione Iniaziale.
Un *Albero Esteso* (ET, *Extended Tree*) di grado k è una struttura dati costituita da un albero binario di ricerca i cui elementi sono delle liste linkate ed ordinate (in senso non decrescente) di valori numerici. Le liste possono contenere al massimo k elementi. Sia ℓ una lista di valori numerici. Indichiamo con $\alpha(\ell)$ la somma degli elementi in essa contenuti, con $\beta(\ell)$ il numero di elementi in essa contenuti e con $\gamma(\ell)$ il valore che si trova in testa alla pila (il più piccolo). Due liste, ℓ_1 e ℓ_2, contenute all'interno dell'Albero Esteso sono ordinate in base ai seguenti criteri:

- se $\alpha(\ell_1) > \alpha(\ell_2)$ allora $\ell_1 > \ell_2$
- se $\alpha(\ell_1) = \alpha(\ell_2)$ e $\beta(\ell_1) > \beta(\ell_2)$ allora $\ell_1 > \ell_2$
- se $\alpha(\ell_1) = \alpha(\ell_2)$, $\beta(\ell_1) = \beta(\ell_2)$ e $\gamma(\ell_1) > \gamma(\ell_2)$ allora $\ell_1 > \ell_2$

Quando un nuovo elemento x viene inserito all'interno di un Albero Esteso, esso va a finire nella lista più piccola contenuta nell'albero, a meno che questa non abbia k elementi. In quest'ultimo caso viene inserita nell'albero una nuova lista vuota nella quale verrà inserito l'elemento x.

1. Si fornisca una classe `C++`, denominata `LList<H>`, che implementi una lista linkata e ordinata, contenente (almeno) i seguenti metodi.
   (a) `LList<H>* ins(H x)` aggiunge un nuovo elemento x alla lista. La funzione restituisce un puntatore ad un oggetto di tipo `LList<H>`;
   (b) `LList<H>* canc(H x)` elimina l'elemento x dalla lista, se presente. La funzione restituisce un puntatore ad un oggetto di tipo `LList<H>`;
   (c) `H* search(H x)` restituisce il puntatore all'elemento x, se esso è presente nella struttura dati. Restituisce `NULL` altrimenti;
   (d) `void print()` è una procedura che stampa in output gli elementi della lista. La stampa procede dall'elemento più piccolo all'elemento più grande della lista.

2. Si fornisca una classe `C++`, denominata `BST<H>`, che implementi un albero binario di ricerca, contenente (almeno) i seguenti metodi.
   (a) `BST<H>* ins(H x)` aggiunge un nuovo elemento x all'albero. La funzione restituisce un puntatore ad un oggetto di tipo `BST<H>`;
   (b) `BST<H>* canc(H x)` elimina l'elemento x dall'albero, se presente. La funzione restituisce un puntatore ad un oggetto di tipo `BST<H>`;
   (c) `H* search(H x)` restituisce il puntatore all'elemento x, se esso è presente nella struttura dati. Restituisce `NULL` altrimenti;
   (d) `void print()` è una procedura che stampa in output gli elementi dell'albero. La stampa procede dall'elemento più piccolo all'elemento più grande della lista.
   (e) `H* minimum()` restituisce il puntatore all'elemento più piccolo dell'albero. Restituisce `NULL` se l'albero non contiene elementi;
   (f) `H* successor(H x)` restituisce il puntatore al successore dell'elemento x, se presente. Restituisce `NULL` se l'elemento x è il più grande tra quelli presenti nella struttura o se x non è presente nella struttura;

3. Si fornisca una classe C++, denominata ET<H>, che implementi un Albero Esteso. Il costruttore della classe dovrà prendere in input il grado k della struttura. La classe dovrà contenere (almeno) i seguenti metodi.
   (a) ET<H>* ins(H x) aggiunge un nuovo elemento x alla struttura dati. La funzione restituisce un puntatore ad un oggetto di tipo ET<H>;
   (b) ET<H>* canc(H x) elimina tutti gli elementi con valore x dalla struttura dati, se presenti. La funzione restituisce un puntatore ad un oggetto di tipo ET<H>;
   (c) void print() è una procedura che stampa in output gli elementi della struttura dati. La stampa procede dalla lista più piccola alla lista più grande dell'albero. per ogni lista la stampa procede dall'elemento più piccolo all'elemento più grande in essa contenuta.

Valutazione.
La prova d'esame verrà valutata anche in base all'ouput generato dal proprio programma su specifici input. Nello specifico lo studente, per la verifica del codice, dovrà eseguire il programma utilizzando il seguente main.

```
int main() {
        /* valutazione del primo esercizio : 8 punti*/
        LList<int>* l = new LList();
        l->ins(3)->ins(7)->ins(1)->ins(8)->ins(2)->ins(4)->print();
        l->canc(3)->canc(9)->canc(5)->canc(1)->ins(10)->ins(5)->print();
        if(l->search(5)) cout << "elemento 5 presente"; else cout << "elemento 5 non presente";
        if(l->search(3)) cout << "elemento 3 presente"; else cout << "elemento 3 non presente";
        /* output:  1 2 3 4 7 8
                    2 4 5 7 8 10
                    elemento 5 presente
                    elemento 3 non presente */
        /* valutazione del secondo esercizio : 10 punti */
        BST<int>* t = new BST();
        t->ins(3)->ins(7)->ins(1)->ins(8)->ins(2)->ins(4)->print();
        t->canc(3)->canc(9)->canc(5)->canc(1)->ins(10)->ins(5)->print();
        if(l->search(5)) cout << "elemento 5 presente"; else cout << "elemento 5 non presente";
        if(l->search(3)) cout << "elemento 3 presente"; else cout << "elemento 3 non presente";
        int *r = t->minimum();
        if(r) cout << "il valore più piccolo è " << *r;
        if(r=t->successor(5)) cout << "il successore di 5 è " << *r;
        if(r=t->successor(3)) cout << "il successore di 3 è " << *r;
        /* output:  1 2 3 4 7 8
                    2 4 5 7 8 10
                    elemento 5 presente
                    elemento 3 non presente
                    il valore più piccolo è 2
                    il successore di 5 è 7 */
        /* valutazione del terzo esercizio : 12 punti */
        ET<int>* b = new ET(3);
        b->ins(3)->ins(8)->ins(6)->ins(10)->ins(9)->ins(5)->print();
        b->ins(11)->ins(13)->ins(18)->ins(2)->ins(1)->ins(4)->print();
        b->canc(11)->canc(13)->canc(18)->canc(9)->ins(14)->print();
        /* output:
                    5 3 6 8 9 10
                    1 2 4 3 6 8 18 9 10 5 11 13
                    1 2 4 3 8 5 14 9 10 */
}
```