Esame di Programmazione del 4/9/2020 Parte di programmazione

Tempo a disposizione 70 minuti + 10 minuti per scan e consegna

Dato un albero binario, lo vogliamo percorrere in larghezza, cioè per livelli (prima la radice, poi i suoi figli, poi i figli dei figli e ogni livello viene percorso da sinistra a destra) producendo una lista L di nodi che puntano ai nodi dell'albero secondo l'ordine determinato dal percorso in larghezza. Quindi il primo nodo di L punta alla radice, il secondo al figlio sinistro della radice (se esiste), il terzo al figlio destro (se esiste) e così via per livelli progressivi dell'albero.

Esempio 1: . Si consideri il seguente albero binario r:

10(5(10(1(_,_),2(_,_)),2(3(_,_),1(_,_))),5(_,1(3(_,_),2(_,_)))) esso contiene 12 nodi.

allora la lista L che vogliamo produrre contiene 12 nodi. Il primo punta alla radice (che ha info=10), il secondo al figlio sinistro (info=5), il terzo al figlio destro (info=5), il quarto al figlio sinistro del figlio sinistro della radice (info=10) e così via.

Per modellare sia alberi binari che liste di nodi che puntano ai nodi di alberi, useremo le seguenti 2 strutture:

struct nodo{int info; nodo* left,*right; nodo(int a=0, nodo*b=0, nodo*c=0){info=a; left=b; right=c;}};

struct nodoE{nodo* info; nodoE* next; nodoE(nodo*a=0, nodoE*b=0){info=a; next=b;}};

Ovviamente la lista L che si desidera costruire è composta da nodi di tipo nodoE.

Si chiede di scrivere una **funzione ricorsiva**, void faiRic(nodoE*&L), che soddisfi le seguenti specifiche:

PRE=(L è una lista ben formata di nodoE che puntano a nodi di un albero binario e la sequenza dei nodi puntati segue l'ordine del percorso in larghezza dell'albero e nessuno dei nodi puntati è discendente di un altro dei nodi presenti, indichiamo con vL il valore iniziale di L)

POST=(alla fine, L sarà la sequenza di nodoE che si ottiene aggiungendo a vL i nodi nodoE che puntano a tutti i discendenti dei nodi puntati da vL ordinati secondo il percorso in larghezza).

Esempio 2: Usando l'esempio precedente, se L inizialmente ha un solo nodo che punta alla radice dell'albero (info=10), alla fine di faiRic L deve essere composta dal nodo originale di L (che punta alla radice) seguito da 11 nuovi nodi che puntano agli altri 11 nodi dell'albero secondo il percorso in larghezza e quindi alla fine i nodi di L puntano ai nodi dell'albero con info: 10,5,5,10,2,1,1,2,3,1,3,e 2. Se invece L inizialmente contenesse un nodo che punta al figlio destro della radice (info=5), alla fine, L dovrebbe contenere 4 nodi che puntano ai nodi con info: 5,1,3 e 2.

Si chiede poi di definire una **funzione iterativa** void filtraIter(nodoE*&L) che prenda la lista L prodotta da faiRic ed elimini i nodi con info ripetuto, mantenendo solo il primo nodo con un dato valore di info. I nodi eliminati vanno deallocati.

Esempio 3: se filtraIter ricevesse la lista L costituita dai 12 nodi nodoE che puntano ai nodi con info, 10,5,5,10,2,1,1,2,3,1,3,e 2, dell'albero, visto che ci sono solo 5 valori distinti tra questi nodi, alla fine del suo calcolo la lista L dovrebbe contenere solo 5 nodi nodoE che puntano ai nodi con gli info distinti, 10,5,2,1,3. Si osservi che la lista finale mantiene l'ordine relativo che i nodi avevano nella lista originale.

Correttezza:

i) Dare la prova induttiva della correttezza di faiRic rispetto alla PRE e POST date.
ii) Descrivere l'invariante del ciclo principale di filtraIter

NB. Entrambe le funzioni richieste possono usare funzioni ausiliarie che dovranno essere ricorsive per faiRic e iterative per filtraIter.