Esame di Programmazione del 1/7/2020 Parte di programmazione

Dato un albero binario R qualsiasi, si chiede di scrivere una funzione ricorsiva che lo percorra in ordine infisso costruendo 2 liste i cui nodi puntano ai nodi dell'albero stesso. Le 2 liste si chiamano *tenere* e *buttare*. Nella lista *tenere* avremo i puntatori a tutti i nodi di R che contengono valori distinti dei campi info. La lista *buttare*, ovviamente, avrà puntatori agli altri nodi di R.

Esempio: se l'albero R contiene 2 nodi con info=5 allora uno dei 2 nodi dovrà essere puntato da un nodo di *tenere* e l'altro sarà puntato da un nodo di *buttare*. Conviene inserire in *tenere* il primo nodo con info=5 che si incontra percorrendo l'albero secondo l'ordine infisso e, successivamente, mettere in *buttare* l'altro nodo con info=5. Se ci fossero molti nodi con info=5 allora il primo di essi, secondo l'ordine infisso, andrà in *tenere* e tutti gli altri andranno in *buttare*.

Se R= 1(**2**(**1**(_,_),_),**3**(2(_,_),2(_,_))) allora *tenere* avrà 3 nodi che puntano ai 3 nodi di R in grassetto, mentre *buttare* punterà agli altri nodi di R. L'ordine dei nodi in *tenere* e *buttare* è irrilevante.

La funzione **ricorsiva** che deve assolvere il compito appena descritto deve avere il seguente prototipo :

void filtra(nodo*R, nodoE*& tenere, nodoE*& buttare)

dove la struttura nodo è la solita che usiamo per i nodi degli alberi binari, mentre nodoE (E sta per Esteso) è la seguente:

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

La funzione filtra avrà probabilmente bisogno di una funzione ausiliaria che determini se un nodo di R ha un campo info uguale a quello di un nodo già visto durante il percorso dell'albero. Anche questa funzione dovrà essere ricorsiva come filtra.

Oltre alla funzione filtra, si chiede di costruire 2 funzioni **iterative**:

i) una funzione iterativa buildBST che usa la lista *tenere* per costruire un albero BST aggiungendo, uno alla volta, i nodi di R puntati dalla lista *tenere*. Inoltre, la funzione, man mano che la lista *tenere* viene consumata, deve deallocare i suoi nodi nodoE non più utili.

ii) la funzione butta che si occupa di deallocare i nodi della lista *buttare*. Attenzione che si tratta di buttare sia i nodi nodoE della lista *buttare* che i nodi di tipo nodo che sono puntati da questi nodoE.

correttezza:

i) scrivere PRE e POST di filtra e anche dell'eventuale funzione ausiliaria usata da filtra.

ii) scrivere l'invariante dei cicli di buildBST.