Scritto di Programmazione del 14/7/2016

Teoria: (3 punti) Specificare una PRE e una POST appropriate per la seguente funzione ricorsiva:

```
int F(nodo* r) {
f(!r->left && !r->right) return 0;
if(!r->left) return 1+ F(r->right);
if(!r->right) return 1+F(r->left);
return F(r->right)+F(r->left);
}
```

Programmazione: **(23 punti)** data una lista concatenata con nodi che hanno campo info intero e tale che ci possano essere nodi distinti con uguale campo info, vogliamo eliminare le eventuali duplicazioni tra nodi, mantenendo, per ogni valore info, l'ultimo nodo della lista originale con quel campo info. Le operazioni da fare sono spiegate nel seguente esempio.

Esempio: sia vL(Q)= 0->1->0->2->1->0->0->2->0, allora vL(Q) deve diventare L(Q) =1->2->0, dove il nodo con 0 è l'ultimo nodo di vL(Q), il nodo con 2 è il penultimo nodo di vL(Q), mentre il nodo con 1 è il quintultimo di vL(Q). Si osservi che questi nodi devono mantenere l'ordine relativo che essi hanno in vL(Q). La lista dei nodi tolti è la lista degli altri nodi di vL(Q) nello stesso ordine relativo che essi avevano in vL(Q) e cioè, 0->1->0->2->0->0.

L'esame consiste di 3 esercizi.

Esercizio 1 (3 punti): scrivere una funzione FIFO push_begin(FIFO x , nodo* b) che inserisce il nodo b all'inizio della lista gestita da x e restituisce il valore FIFO che gestisce la lista così ottenuta. La struttura FIFO è specificata nel programma che avete a disposizione ed è costituita da 2 campi di tipo nodo* che si chiamano primo ed ultimo e che puntano, rispettivamente, al primo e all'ultimo nodo di una lista. Il programma dato contiene la funzione push_end che aggiunge un nodo alla fine della lista gestita da un valore FIFO.

Esercizio 2 (10 punti): scrivere una funzione **ricorsiva** che esegua le operazioni descritte nell'esempio precedente. La funzione deve avere il seguente prototipo FIFO tieni_ultimo_ric(nodo*&Q) e deve essere corretta rispetto alle seguenti PRE e POST.

PRE=(L(Q) è una lista corretta e vL(Q)=L(Q))

POST=(L(Q) è ottenuta da vL(Q) eliminando i nodi con info ripetuto mantenendo solo l'ultimo nodo per ciascun campo info e mantenendo l'ordine relativo che questi nodi hanno in vL(Q). Inoltre restituisce un valore FIFO f tale che f.primo è la lista dei nodi eliminati nello stesso ordine relativo che essi hanno in vL(Q))

Nota: La funzione push_begin dell'esercizio (1) dovrebbe tornare utile per scrivere tieni_ultimo_ric.

Esercizio 3 (10 punti): si richiede di risolvere lo stesso problema del punto(2), ma con una funzione **iterativa** che abbia il seguente prototipo, FIFO tieni_ultimo_iter(nodo*& Q) e tale che sia corretta rispetto alla stesse PRE e POST dell'esercizio (2).

Osservazione:sia per l'esercizio (2) che per il (3), l'ordine richiesto per la lista dei nodi tolti suggerisce (e forse impone) di esaminare la lista L(Q) da sinistra a destra, sia nel caso ricorsivo che in quello iterativo.

Avviso: tutte le funzioni ausiliarie che eventualmente introdurrete dovranno essere ricorsive per la parte (2) e iterative per la (3). Nessun nuovo nodo deve essere creato dalle vostre funzioni, né deallocato.

Correttezza:

1) **(5 punti)** Delineare la prova induttiva di correttezza di tieni_ultimo_ric rispetto alla PRE e POST date.

2) **(3 punti)** Descrivere l'invariante del ciclo principale (quello più esterno) della funzione tieni_ultimo_iter.