Esercizio 2 del 3/5/2021

Il problema da affrontare è facile da spiegare: data una lista concatenata ed un valore z, si vuole eliminare dalla lista tutti i nodi che hanno campo info uguale a z. Si chiede di realizzare questa operazione in 3 modi diversi che corrispondono alle 3 tipologie di soluzioni ricorsive viste a lezione per il problema di aggiungere un nodo in fondo ad una lista. Le 3 funzioni devono soddisfare le seguenti specifiche:

PRE_1=(Lista(L) ben formata, vLista(L)=Lista(L))

nodo* del1(nodo* L, int z)

POST_1=(restituisce una lista composta dai nodi di vLista(L) che non hanno campo info=z nell'ordine che hanno in vLista(L)) && (dealloca gli altri nodi di vLista(L))

PRE_2=(LIsta(L) ben formata, non vuota, e L->info != z, sia vLista(L)=Lista(L))

void del2(nodo*L, int z)

POST_2=(Lista(L) è ottenuta da vLista(L) eliminando i nodi con info=z e tenendo gli altri mantenendo il loro ordine) &&(i nodi con info=z sono deallocati)

PRE_3=(Lista(L) ben formata, vLista(L)=Lista(L))

void del3(nodo*&L, int z)

POST_3=(Lista(L) è ottenuto da vLista(L) eliminando i nodi con info=z e mantenendo gli altri nello stesso ordine che avevano in vLista(L) && (i nodi eliminati sono deallocati)

Del1 segue lo schema più semplice in cui la funzione riceve la lista iniziale e restituisce la nuova lista col return. Del1 esegue delle operazioni inutili perché al ritorno collega tra loro nodi che potrebbero già essere collegati. Del2 evita le operazioni inutili eliminando i nodi con info=z partendo dal nodo immediatamente precedente. Ovviamente è necessario garantire che ogni volta che si deve eliminare un nodo, questo sia preceduto da un nodo che invece va tenuto (info != z). Per garantire questa proprietà, basta garantire che la PRE_2 data sia vera prima di ogni invocazione ricorsiva. La PRE_2 richiede che la lista in ingresso non sia mai vuota e che il suo primo nodo non abbia info uguale a z. Insomma il primo nodo sia da tenere. Da quel nodo si partirà ad esaminare se il secondo nodo è da buttare e a seconda del caso lo si eliminerà collegando il primo al terzo nodo, oppure se il secondo è da tenere. si passerà a lui per esaminare il terzo nodo, e così via. Finalmente del3 non farà operazioni inutili come del 2, ma, al contrario di del2, non richiederà condizioni particolari sui nodi della lista grazie al passaggio di L per riferimento.

Notare che le POST richiedono tutte che i nodi tolti dalla lista originale vengano deallocati.
Viene dato un main e le funzioni di i/o. Si chiede di non modificare il main.

Correttezza: fare le dimostrazioni induttive di correttezza deile 3 funzioni.