Prova di correttezza di no_rip dell'esame del 30/6/2015

La funzione ricorsiva no_rip è come segue:

```
nodo* no_rip(nodo*L)
{
  if(L)
 {
   L->next=togli(L->next, L->info);
   no_rip(L->next);
   return L;
 }
 return 0:
}
```

Visto che no_rip usa la funzione togli, occorre ricordare, dal testo del compito, la PRE e POST di togli. Dopo ricordiamo anche PRE e POST di no_rip. Nella prova di no_rip assumeremo che togli sia corretta rispetto alle sue PRE e POST. Naturalmente questo fatto dovrebbe essere provato a parte.

PRE=(lista(C) è corretta e y è definito)

nodo* togli(nodo*C, int y)

POST=(restituisce la lista (ovviamente corretta) che resta eliminando da C i nodi con campo info=y. I nodi tolti da C sono deallocati. Nessun nodo nuovo rispetto a C è allocato)

PRE=(lista(L) è corretta, L=vL)

nodo* no_rip(nodo*L)

POST=(restituisce quello che resta di vL eliminando tutti quei nodi che sono preceduti da un nodo con uguale campo info. I nodi tolti da vL sono deallocati. Nessun nuovo nodo rispetto a vL è allocato)

Prova induttiva di no_rip:

caso base: L=0: nessun nodo e quindi niente da fare. Restituire 0 soddisfa la POST.

passo ricorsivo: L contiene almeno un nodo: allora la funzione invoca togli(L->next, L->info). L'invocazione soddisfa la PRE di togli infatti, visto che L !=0, L->next è lista corretta. Per lo stesso motivo L->info è definito. Quindi possiamo assumere che al ritorno di togli valga la sua POST e cioè che venga restituito quello che resta di L->next una volta eliminati tutti i nodi con campo info=L->info. Chiamiamo L' questa lista. Inoltre la POST di togli garantisce che i nodi tolti da L ->next sono stati deallocati. L' è puntata dal nodo L e quindi ora la lista L non contiene altri nodi, oltre L stesso, con campo info= L->info.

(+)Si osservi che i nodi di L' sono nodi di vL e che l'ordine tra questi nodi in L' è lo stesso che in vL.

A questo punto viene effettuata l'invocazione ricorsiva no_rip(L->next). La PRE_RIC è verificata, ricorda che L->next è L' che è restituita da togli e quindi è una lista corretta (ovviamente) . Possiamo quindi assumere POST_RIC= (restituisce quello che resta di L' eliminando tutti quei nodi che sono preceduti da un nodo con uguale campo info. I nodi tolti da L' sono deallocati. Nessun nuovo nodo rispetto a L' è allocato). Sia L'' la lsita prodotta da no_rip. Dalla POST_RIC, L'' non ha nodi con uguale campo info e per ogni valore dei campi info di L', conserva il primo nodo in L' con quel campo info. Per l'osservazione (+) i nodi di L'' avranno

lo stesso ordine che avevano in vL. Basta a questo punto osservare che L è il primo nodo di vL e ricordare che L' non ha nodi con info=L->info, per vedere che L@L'' è la lista che soddisfa la POST e che infatti viene restituita dalla funzione no_rip. La deallocazione dei nodi è garantita da POST di togli e da POST_ric.

Resta un piccolo punto ancora da chiarire: siamo sicuri che L punti alla lista L'' restituita dall'invocazione ricorsiva no_rip(L->next)? Dobbiamo considerare 2 casi:

a) L' = 0, il che significa che togli ha eliminato tutti i nodi che seguivano L o perché non c'è alcun nodo (L ha solo il primo nodo) oppure perché tutti i nodi hanno lo stesso campo info=L->info. In questo caso L->next=togli(L->next, L->info); inserisce 0 nel campo next di L.
Ovviamente in questo caso l'invocazione ricorsiva no_rip(L->next) restituisce 0. Quindi la funzione no_rip restituisce il solo nodo L il che è corretto rispetto alla POST.

b) L' !=0 e allora L'' restituita da no_rip(L') avrà sicuramente lo stesso primo nodo di L' (esso è il primo nodo in L' col suo valore di info). Quindi la riga, L->next=togli(L->next, L->info); garantisce che L sia collegato a L'' e quindi basta eseguire no_rip(L->next); e non serve L->next = no_rip(L->next); (anche se non sarebbe sbagliato).