Esercizio 2 del 26/3/2013 da consegnare corretto entro il 3/3 a mezzanotte

Viene dato un programma con la definizione della struttura nodo, una funzione che stampa liste concatenate e un main che compie varie operazioni sui file "input" e "output" e che invoca la funzione void crea(nodo* &L, int dim, ifstream & INP), simile (ma non uguale!) che costruisce una lista di dim nodi che hanno come campo informativo interi letti da "input". La funzione crea deve soddisfare la seguente pre- e post-condizione:
PRE=("input" contiene (almeno) dim interi a_1,..,a_dim, dim>=0)
void crea(nodo*& L, int dim, ifstream & INP)
POST=(la funzione restituisce nella variabile L, passata per riferimento, una lista corretta di dim nodi tali che il primo nodo contiene nel campo info a_1, il secondo nodo contiene a_2, e l'ultimo contiene a_dim)

Esempio: *se dim=3 e si leggono gli interi 2, 3 e 4, crea deve costruire e restituire nel parametro L una lista (corretta) di 3 nodi con campi informativi 2, 3 e 4 (dove 2 è il campo info del primo nodo delle lista e 4 quello dell'ultimo).*

Dopo queste operazioni, il main legge da "input" l'intero dimP (0<dimP<=20) e legge i successivi dimP interi nell'array int P[20]. A questo punto il main invoca una funzione nodoG* G(nodo*&L, int*P, int dimP). Il tipo nodoG è spiegato nell'esempio che segue. La scrittura di G è la parte principale dell'esercizio. Il compito di G è di cercare un match, anche non contiguo e anche non completo di P nella lista L. Spieghiamo con un esempio in cui introduciamo anche notazioni utili per scrivere POST_G:

Esempio: *supponiamo che L= 2->3->3->1->4 e che P=[3,1,2] con dimP=3. Allora il massimo match di P in L è costituito dal secondo e dal quarto nodo di L. G deve restituire quello che resta di L una volta estratti da L i nodi del match, quindi nel nostro esempio, dovrà restituire col parametro L (passato per riferimento!) la lista 2->3->4. Nel seguito indicheremo questa lista con (L-P). I nodi 3 e 1, staccati da L, devono venire restituiti attraverso una lista costituita da nodo di tipo nodoG come segue: struct nodoG{nodo* N, nodoG* next;};. Nel nostro esempio, G, con il return, dovrà restituire una lista corretta di 2 nodi di tipo nodoG, dove il campo N del primo punta al nodo 3 (estratto da L) e il secondo nodo ha il campo N che punta al nodo 1 (anch'esso estratto da L). Nel seguito chiameremo questa lista Comp(L-P).*

Si noti che il match dell'esempio non è contiguo (ma in generale nulla vieta che lo sia), che il match non deve necessariamente essere completo (nell'esempio si trova il match solo dei primi 2 elementi di P e non del terzo), ma naturalmente il match deve essere il più lungo possibile. Osservare anche che i nodi di Comp(L-P) conservano l'ordine relativo che i nodi estratti da L avevano nella lista originale. Nel nostro esempio il nodo di Comp(L-P) che punta a 3 è il primo nodo e quello che punta a 1 è il secondo. Osservare inoltre che ogni elemento di P deve venire matchato al piu' presto possibile in L. Per esempio il primo elemento di P viene matchato con il secondo nodo della lista e non col terzo. In questo modo la definizione di (L-P) e di Comp(L-P) è sempre unica. Si osservi infine che, qualora (L-P) = L, Comp(L-P) sarebbe vuota, se invece (L-P) fosse vuoto, allora Comp(L-P) avrebbe tanti nodi quanti ne aveva L e punterebbe a tutti i nodi di L nell'ordine che essi avevano in L.

La funzione G deve soddisfare la seguente pre- e post-condizione.

PRE_G=(L è lista corretta e L=vL, P ha dimP elementi definiti con dimP>=0)

POST_G=(G restituisce col return Comp(vL-P) e (vL-P) attraverso il parametro L passato per riferimento).

Dopo l'invocazione di G, il main stampa opportunamente i nodi delle 2 liste prodotte da G. Si richiede la dimostrazione induttiva di correttezza della funzione G.

Attenzione: G non deve creare né distruggere alcun nodo di tipo nodo, mentre, se Comp(L-P) non è vuoto, dovrà creare qualche nodo nodoG.