Esercizi del 7/3/2013

INFORMAZIONI IMPORTANTI: Il programma da fare deve risiedere sul file esercizio1.cpp e deve leggere l'input da input1 e scrivere l'output su output1. Il comando di consegna è consegna esercitazione.

Teoria: (i) Data la seguente funzione ricorsiva, inserire appropriate PRE e POST. Per ottenere una POST semplice è necessario inserire nella PRE delle condizioni su L1 ed L2.

```
//PRE= ??
nodo* F(nodo* L1, nodo*L2) {
 if (!L1) return L2;
 if (!L2) return L1;
 if (L1->info >=L2->info) {L1->next=F(L1->next,L2); return L1;}
    else
   {L2->next=F(L1,L2->next); return L2;}
} //POST=??
```

(ii) che potete dire sul seguente programma?

```
void G(int x){cout<<x+y<<endl;}
int y=10;
main(){int y=20,x=30; G(x); }
```

(iii) e su questo programma?

```
void G(int x){…}
int G(char x){….}
main(){double y=20; G(y); }
```

Programmazione: nel file esercizio1.cpp viene dato un programma che deve essere completato. Nel programma, c'è la definizione della struttura nodo, c'è una funzione che stampa liste concatenate e c'è un main da completare che compie varie operazioni di input e output e che invoca una funzione nodo* crea(int dim, ifstream & INP). La funzione crea non è data e scriverla è parte dell'esercizio. La funzione crea deve leggere dim interi da INP e contemporaneamente deve costruire una lista di dim nodi che hanno come campo informativo gli interi letti. Nel seguito chiameremo L il puntatore al primo nodo della lista restituita da crea.

Esempio: *se dim=3 e si leggono gli interi 2, 3 e 4, crea deve costruire e restituire col return 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 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 nodo* F(nodo*&L, int*P, int dimP). La scrittura di F è la parte principale dell'esercizio. Il compito di F è di cercare un match, anche non contiguo di P nella lista L. Spieghiamo con un esempio in cui introduciamo anche notazioni utili per scrivere POST_F:

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. Con L-P indicheremo 2->3->4, cioè i nodi di L che restano dopo aver tolto quelli che partecipano al match, mentre chiameremo R(L,P) la lista 3->1 che contiene i 2 nodi estratti da L. 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 che naturalmente il match deve essere il più lungo possibile. Osservare anche che i nodi di R(L,P) conservano l'ordine relativo che avevano nella lista originale. Osservare inoltre che ogni elemento di P viene 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 R(L,P) è unica. Si osservi infine che R(L,P) potrebbe essere vuota, nel qual caso L-P = L e che può anche succedere che R(L,P)=L nel qual caso L-P sarebbe vuoto.*

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

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

POST_F=(F restituisce col return R(vL,P) e vL-P attraverso il parametro L passato per riferimento).

Dopo l'invocazione di F, il main stampa le 2 liste prodotte da F invocando 2 volte la funzione stampa che è data.

Si richiede la dimostrazione induttiva di correttezza della funzione F.

Attenzione: F non deve né creare alcun nuovo nodo, né distruggere alcun nodo esistente.