Esercizi del 28/2/2013

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: Il seguente programma è corretto o no? Spiegate la vostra risposta e, se pensate sia corretto, spiegate cosa stampa e perché:

```
int ** g(int ** x){int *p=*x+2; x=&p; return x;}

main() {int X[]={1,2,3}, *q=X-1; int*& z= *g(&q);  *z=X[2]; cout<<X[0]<<X[1]<<X[2]<<endl;}
```

Programmazione:

Esercizio 1: Si tratta di un problema di pattern match tra un array int T[dim] con dim elementi definiti e un pattern int P[dimP]. Si richiede di realizzare 2 funzioni entrambe **ricorsive**.

La prima funzione si chiama F ed ha il seguente prototipo: int F(int*T, int dim, int *P, int dimP) che rispetta le seguenti PRE_F e POST_F:

PRE_F=(T ha dim elementi definiti, P ha dimP elementi definiti)

POST_F=(F restituisce il n. di match di P in T considerando anche match sovrapposti)

Esempio: *Sia T=[3,5,3,5,3,5,3,5] e P=[5,3,5]. Ci sono 3 match di P in T che iniziano nelle posizioni 1, 3 e 5 di T e tali che il primo e il secondo match si sovrappongono nella posizione 3 di T, mentre il secondo e il terzo sono sovrapposti nella posizione 5 .* Quindi F dovrebbe restituire l'intero 3.

La funzione F deve invocare una funzione ricorsiva bool match(int* T, int dim, int*P, int dimP) che soddisfa la seguente coppia PRE_match e POST_match:

PRE_match=(T da dim elementi definiti, P ha dimP elementi definiti)

POST_match=(match restituisce true sse c'è un match (completo e contiguo) di P in T a partire da T[0]).

E' necessario scrivere anche un main che apre i file input1 e output1 e legge da input1, dim (1<= dim <=100), seguito da dim valori da mettere in int T[100], poi dimP (1<=dimP<=20), seguito da dimP valori da mettere in int P[20]. Il main dovrà invocare F e scrivere su output1 l'intero restituito da F.

Da fare: iniziate a realizzare la funzione match che è più semplice. Fate la prova induttiva di match rispetto a PRE_match e POST_match date. Poi se riuscite fate anche F con la sua prova induttiva.

Esercizio 2: si tratta di modificare la funzione F dell'esercizio 1 in modo che conti i match non sovrapposti di P in T . PRE_F e POST_F sono gli stessi di prima con la richiesta aggiuntiva di match non sovrapposti in POST_F. Nel caso dell'esempio precedente la risposta della nuova F sarebbe 2, essendoci 2 match non sovrapposti (il primo e il terzo dell'esempio).

Test: i test automatici verranno fatti solo sulla funzione F dell'Esercizio 1. Quindi il main deve stampare su output1 solo il risultato di F dell'Esercizio 1.