Compito di Programmazione del 2 settembre 2011

Teoria:

(i) Data la seguente funzione ricorsiva, inserire appropriate PRE e POST. Per ottenere una POST semplice è necessario inserire nella PRE delle condizioni sui valori di 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) Considerate il seguente programma. In caso pensiate sia corretto, spiegate cosa calcola e cosa stampa. Se pensate sia sbagliato, spiegate in dettaglio il motivo. In ogni caso, per spiegare le vostre conclusioni, disegnate uno schema dettagliato che mostri la relazione tra le variabili e i puntatori in gioco.

```
int* f(int **p){int b=3,*x=&b; x=(*p)+1; return x-2; }

main() {int b[]={1,2,3,4},*q=b+2; *f(&q)=*q; cout<<b[0]<<b[1]<<b[2]<<b[3];}
```

Programmazione: il problema da considerare è lo stesso sia per la parte iterativa che per quella ricorsiva. Si tratta di trovare un match di un pattern int P[n] in un testo int T[n][n] nel modo seguente. P[0] viene cercato sulla colonna 0 di T, se viene trovato in posizione T[k0][0], allora si cerca P[1] nella colonna 1 a partire dalla posizione k0, cioè nelle posizioni T[k0..n-1][1]. Se P[1] viene trovato in T[k1][1], allora si cercherà P[2] nella porzione della terza riga, T[k1..n-1][2] e così via. Se la ricerca di un P[i] fallisce, allora il match fallisce. Se invece tutti gli n elementi di P vengono trovati nelle n colonne di T nel modo descritto prima, allora il match ha successo ed un tale match viene descritto da un array int R[n] tale che i suoi valori sono indici di righe, cioè compresi tra 0 e n-1 e, per quanto detto prima, R deve soddisfare anche la condizione che per ogni $j \in [0,n-2]$, R[j]<=R[j+1].

Esempio. Sia n=3, P= [8,8,10] e T=[7,9,2][8,8,4][10,8,10][1]. Allora esiste un match di P in T che è descritto dall'array R=[1,1,2]. Anche R'=[1,2,2] descrive un altro match. Se P fosse [8,9,10] non ci sarebbe match di P in T.

Parte iterativa. Si chiede di sviluppare una funzione iterativa int*F(int*T,int n, int*P) che restituisca un array che descriva un (qualsiasi) match di P in T se un tale match esiste, mentre se nessun match di p in t esiste, allora deve restituire 0. E' consigliabile che F usi una funzione iterativa ausiliaria. Si osservi che T, sebbene vada trattato come un array n x n, viene passato come un array ad una sola dimensione.

1) Scrivere PRE e POST per F

2) Scrivere la funzione F e l'eventuale funzione ausiliaria.

3) Scegliere un ciclo significativo di F e specificare un invariante che descriva appropriatamente cosa "fa" il ciclo oppure, nel caso di uso della funzione ausiliaria, specificare le sue PRE e POST.

---

[1] T è un array 3 x 3. Per convenienza, le sue 3 righe sono scritte una di seguito all'altra.

Parte ricorsiva: Si chiede di sviluppare una funzione ricorsiva int* Fric(int*M, int n, int*P,.....) che restituisca un array R che descriva un match di P in T se esiste e altrimenti 0. I puntini nella lista dei parametri indicano che si può (e infatti si deve) aggiungere parametri. E' consigliabile che Fric usi una funzione ausiliaria anch'essa ricorsiva.

1) Scrivere PRE e POST di Fric, tenendo conto che si tratta di una funzione ricorsiva e che quindi PRE e POST devono avere senso per ogni invocazione di Fric e non solo per la prima invocazione.

2) Realizzare Fric e l'eventuale funzione ausiliaria.