## Compito di Programmazione del 4 luglio 2011

Teoria:

(i) Data la seguente funzione ricorsiva, inserire appropriate PRE e POST

```
//PRE= ??
int F(Nodo* a) {
If(!a) return 0;
 if (!a->left && !a->right) return 1;
 return -1+F(a->left)+F(a->right);
} //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. Si ricorda che la dereferenziazione ha precedenza sulla somma.

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

**Programmazione**: Sia la parte iterativa che quella ricorsiva sono ispirate dallo stesso problema che concerne la gestione di una lista di liste in modo da realizzare un'operazione di eliminazione dei nodi con certe caratteristiche. Vediamo un esempio.

**Esempio**. *Supponiamo di avere una lista di liste LL che consiste di 3 nodi, Na, Nb e Nc che puntano alle seguenti liste di interi:*
*Na->4->2->1*
*Nb->2->2*
*Nc->3*
*Immaginate di volere eliminare i nodi che contengono 2. Ovviamente si tratta di eliminare un nodo dalla lista puntata da Na, di eliminare tutti i nodi da quella puntata da Nb, mentre quella di Nc resterebbe inalterata visto che non contiene nodi con campo informativo 2.*
*Visto che la lista puntata da Nb sarebbe vuota, il nodo Nb dovrebbe venire eliminato e quindi la nuova LL diventerebbe:*
*Na->4->1*
*Nc->3*

Le liste di liste sono costituite da nodi del seguente tipo struttura NN: struct NN{nodo* lista; NN*next;};, mentre i nodi che contengono interi sono realizzati con struct nodo{int info; nodo*next;};.
**L'eliminazione di un nodo presuppone la sua deallocazione dallo heap** (cioè la memoria in cui sempre risiedono i valori allocati dinamicamente).

**Esercizio iterativo:**
si tratta di realizzare una funzione iterativa G come segue:
a) G ha prototipo nodo* G(nodo* L, int x) e deve soddisfare la seguente PRE e POST:
PRE=(L è una lista ben formata[1] di valore K e x un intero)
POST=(G restituisce col return quello che resta di K dopo aver eliminato i nodi che contengono x, i nodi eliminati vengono deallocati).
b) Si chiede si scrivere l'invariante del ciclo principale della funzione G e di delineare una prova di correttezza della funzione stessa;

[1] Ben formata significa che la lista L è vuota oppure termina con un nodo con campo next uguale a 0

## Esercizio ricorsivo:

Si tratta di realizzare 2 funzioni ricorsive Fric e Gric come segue:
a) Fric ha prototipo void Fric(NN*& LL, int x) e deve soddisfare la seguente PRE e POST:
PRE=(LL è una lista di liste ben formata di valore K e x un intero)
POST=(alla fine dell'esecuzione di Fric il valore di LL è diventato quello ottenuto da K dopo aver eliminato i nodi di tipo nodo che contengono x ed i nodi di tipo NN che risultino puntare a liste vuote dopo le eliminazioni precedenti, i nodi eliminati vengono deallocati).
b) Fric deve usare la seconda funzione richiesta Gric che deve avere il seguente prototipo: void Gric(nodo* &L, int x) e che ha il compito di eliminare ricorsivamente da L i nodi che contengono x;
c) Si chiede si scrivere una PRE ed una POST per Gric e di realizzare la funzione ricorsiva Gric stessa;