**Prova d'esame di Laboratorio di Programmazione Strutturata
per il corso di laurea in Scienze e Tecnologia dei Media
Prova scritta – 23 Settembre 2013**

• **Esercizio 1**

Si scriva una *function ricorsiva* che effettua il calcolo di un qualsiasi elemento F_n della successione di Fibonacci, per ogni indice intero $n \geq 0$. Per fissare le idee, il prototipo della *function* richiesta è il seguente:

```
int fibo_ricor(int n);
```

inoltre, si ricordi che una *function* è detta *ricorsiva*, quando al suo interno essa richiama se stessa. Infine, si riporta qui di seguito la definizione della successione di Fibonacci nella sua classica notazione matematica:

$$F_0 = F_1 = 1, \quad F_n = F_{n-1} + F_{n-2} \quad \forall n \geq 2.$$

All'interno di un commento della *function* `fibo_ricor`, si indichi chiaramente come varia la sua *complessità computazionale* in funzione del valore di `n`.

• **Esercizio 2**

Si scriva una *function* che, in modo *iterativo*, effettua il calcolo di un qualsiasi elemento F_n della successione di Fibonacci, per ogni indice intero $n \geq 0$. Per fissare le idee, il prototipo della *function* richiesta è il seguente:

```
int fibo_iter(int n);
```

si suggerisce di utilizzare almeno tre variabili `Fr`, `Fu`, `Fp` e un contatore `j`, all'interno di tale *function*. Infatti, è consigliabile inizializzare opportunamente le variabili `Fr`, `Fu` e `Fp` per poi realizzare un ciclo sul contatore `j`, in modo tale che ad ogni iterazione il valore di `Fr` sia ridefinito così da essere uguale a $F_{j-1} + F_{j-2}$, mentre i valori di `Fp` e `Fu` devono essere posti uguali a, rispettivamente, F_{j-1} e F_j . In altri termini, le variabili `Fp` e `Fu` assumono rispettivamente il significato di penultimo e ultimo valore calcolato della successione di Fibonacci; infine, *al termine dell'esecuzione del ciclo* sul contatore `j`, in `Fr` sarà contenuto il valore richiesto (cioè proprio F_n).

Anche per quanto riguarda la *function* `fibo_iter`, all'interno di un commento, si indichi chiaramente come varia la sua *complessità computazionale* in funzione del valore di `n`.

• **Esercizio 3**

Si consideri una *lista concatenata bidirezionale* che è implementata in linguaggio `C`, in modo tale che ogni suo nodo è rappresentato da una `struct` costituita da tre campi: il primo è di tipo `int`, mentre il secondo e il terzo sono due *puntatori* alle strutture che indirizzano rispettivamente al nodo successivo e a quello precedente della sequenza costituita dalla lista stessa. Per fissare le idee, ogni nodo di tipo `NodoLista` sia definito come segue:

```

struct NODO {
    int info;
    struct NODO *next;
    struct NODO *prev;
};
typedef struct NODO NodoLista;

```

Come al solito, la *lista concatenata bidirezionale* può essere ricostruita a partire da un *puntatore* al primo elemento della sequenza. È quindi utile introdurre il tipo `Lista` definito come segue:

```

typedef struct NodoLista* Lista;

```

Inoltre, come da tradizione, l'ultimo elemento della lista è quel nodo tale che il suo puntatore `next` contiene l'indirizzo (fasullo) `NULL`. Il secondo puntatore costituisce una struttura di tipo *circolare*, ciò significa che il primo elemento punta all'ultimo, il quale a sua volta indirizza al penultimo e così via fino al secondo elemento, il quale punta al primo elemento della lista, completando la circolarità.

Si scriva una *function* il cui prototipo è il seguente:

```

void stampa_allindietro(Lista L);

```

essa si limita a visualizzare sullo schermo i nodi della lista (o, più esattamente, le informazioni relative ai nodi stessi) in ordine inverso. In dettaglio, le scritte che devono apparire sul video sono del tipo seguente:

```

Il nodo con info u non precede alcun altro nodo
Il nodo con info p precede il nodo con info u

```

...

```

Il nodo con info s precede il nodo con info t

```

```

Il nodo con info p' precede il nodo con info s

```

Ovviamente, nell'elenco precedente la variabile *u* rimpiazza il valore del campo `info` relativo all'ultimo nodo, mentre *p* è quello relativo al penultimo nodo e così via fino a *t*, *s* e *p'* che rappresentano i valori del campo `info` contenuti, rispettivamente, nel terzo, nel secondo e nel primo nodo.

Nella codifica della *function* `stampa_allindietro`, si consiglia di prestare particolare attenzione ai seguenti due casi:

- la lista è costituita da un solo nodo che è quello che si trova all'indirizzo di memoria `L`;
- la lista è vuota. In quest'ultimo caso si faccia apparire sullo schermo la scritta seguente:

```

Non ci sono nodi nella lista!

```