# Esame di Programmazione
# 1 febbraio 2011
# Corso di Laurea in Scienze e Tecnologie Informatiche
# A.A. 2010/2011

Tempo a disposizione: 1,5 h

**Esercizio 1.**

Realizzare una implementazione del concetto di coda basata sulla seguente idea:

- La struttura dati coda è costituita da due stack, A e B;
- L'aggiunta di un elemento alla coda si effettua mediante *push* sullo stack B;
- Il prelievo di un elemento dalla coda si effettua mediante *pop* dallo *stack* A; se lo *stack* A è vuoto, prima di fare il prelievo si trasferiscono tutti gli elementi dallo *stack* B allo *stack* A (mediante push e e pop);
- La coda si considera vuota se e solo se sono vuoti entrambi gli *stack* A e B;

**Definire** la struttura dati stack realizzata tramite array dinamico. **Implementare** poi le funzioni InCoda, OutCoda e CheckCodaVuota che operino come precedentemente descritto.

**Esercizio 2.**

Considerando il codice riportato sotto, disegnare e commentare le strutture dati create arrivati al PUNTO A e al PUNTO B, indicare inoltre cosa viene stampato a video rispettivamente dalle due chiamate alla funzione funzione2()

```
typedef struct {
    int valore;
    struct Cella *next;
} Elemento;

typedef Elemento *PElemento;

void funzione1(PElemento *T, int valore);
void funzione2(PElemento T);
PElemento funzione3(PElemento T);

int main() {
    PElemento testa;
    testa = NULL;

    funzione1(&testa, 7);
    funzione1(&testa, 3);
    funzione1(&testa, 6);
    funzione1(&testa, 1);
    funzione1(&testa, 4);

    funzione2(testa);

/* PUNTO A */

    testa = funzione3(testa);

/* PUNTO B */
```

```
    printf("\n");
    funzione2(testa);

    system("PAUSE");
    return 0;
}

void funzione1(PElemento *T, int valore) {
    PElemento n, tmp, pr;
    n = (PElemento)malloc(sizeof(Elemento));
    n->next = NULL;
    n->valore = valore;
    if (*T == NULL) {
        *T = n;
    }
    else {
        pr = NULL;
        tmp = *T;
        while (tmp != NULL) {
            if (tmp->valore > n->valore) {
                if (pr == NULL) {
                    *T = n;
                }
                else {
                    pr->next = n;
                }
                n->next = tmp;
                return;
            }
            pr = tmp;
            tmp = tmp->next;
        }
    }
}

void funzione2(PElemento T) {
    if (T == NULL) {
        return;
    }
    printf("%d ", T->valore);
    funzione2(T->next);
}

PElemento funzione3(PElemento T) {
    PElemento tmp;

    if (T == NULL) {
        return NULL;
    }
    if (T->valore % 2 == 1) {
        T->next = funzione3(T->next);
        return T;
    }
    else {
        tmp = funzione3(T->next);
        T->next = NULL;
        free(T);
        return tmp;
    }
}
```