

## Fondamenti di Informatica A-L – (Prof. P. Zingaretti)

PROVA SCRITTA – 10 luglio 2003

### Avvertenze:

- \* Consegnare **solo fogli formato A4**
- \* **Scrivere su un solo lato** (no fronte-retro)
- \* In ordine di preferenza usare: 1) **inchiostro nero**; 2) matita; 3) inchiostro rosso; 4) inchiostro blu
- \* **In testa a ciascun foglio** scrivere: cognome, nome, numero progressivo di pagina rispetto al totale  
esempio per il secondo foglio di 3 consegnati: Giuseppe Russo 2/3
- \* Mantenere sul banco il **libretto o altro documento di riconoscimento** fino a controllo avvenuto
- \* Nient'altro deve trovarsi sul banco: **non è consentito consultare libri, dispense, appunti, ecc.**
- \* La **correzione** di riferimento per l'autovalutazione verrà effettuata in questa stessa aula alle ore **15:45**
- \* La consegna delle **fotocopie** dei compiti avverrà presso l'Istituto di Informatica al termine della correzione
- \* La **prova orale** si terrà **giovedì 17 luglio** alle ore **9:00** presso l'Istituto di Informatica



Si ricorda che chi si presenterà all'orale **DEVE** portare l'implementazione al computer della propria soluzione, eventualmente corretta, **corredata di tutto quanto necessario alla verifica** del corretto funzionamento.

### 1. (7 punti)

Si definisca una funzione C (Pascal) che riceve in input una matrice di interi con N righe e M colonne (N e M qualsiasi) e restituisce sia il numero di tutti gli elementi della matrice con valore pari che il numero di tutti gli elementi della matrice con valore dispari.

### 2. (11 punti)

Si definisca una funzione C (Pascal) che riceve in input la rappresentazione binaria di un numero e restituisce la sua rappresentazione esadecimale.

Suggerimento: sfruttare il fatto che una cifra esadecimale corrisponde a 4 cifre binarie.

### 3. (12 punti)

Si scriva una funzione C (Pascal) che inserisce un intero in modo ordinato su una lista dinamica (rappresentazione collegata mediante puntatore) bidirezionale.

### 1. (7 punti)

```
void even_odd(int *MAT, int N, int M, int *even, int *odd) {
    int i, j;
    *even=*odd=0;    //azzero
    for (i=0; i<N; i++)
        for (j=0; j<M; j++)
            if (*(MAT+i*M+j) %2) (*even)++;
            else (*odd)++;
}
```

### 2. (11 punti)

```
char *bin2esa(char *bin) {
    int a, h=0, i=0, j, k=0, n, val=0, A[4];
    char *s;
    //calcolo e allocazione stringa output: 3 punti
    while (bin[i]) i++;    //i=strlen(bin)=lunghezza stringa binaria
    j=i/4;                //4 cifre binarie = 1 cifra esadecimale
    if (i%4) j++;        //j=numero totale cifre esadecimali
    s=(char *)malloc((j+1)*sizeof(char)); //stringa esadecimale
    //scrittura stringa output: 6 punti
    if (i%4) { //cifra esa + significativa con meno di 4 cifre binarie
        for (a=0; a<4; a++) A[a]=0; //azzero
        for (j=4-i%4; j<4; j++, k++)
            if (bin[k]=='1') A[j]=1; //k contatore cifre binarie processate
        val=8*A[0]+4*A[1]+2*A[2]+A[3]; //A[0]<<3+A[1]<<2+A[2]<<1+A[3]
        s[h++]=val + (val > 9 ? 'A'-10:'0'); //h contatore cifre esa
    }
    for (n=0; n<i/4; n++) { //altre cifre esa di 4 cifre binarie
        for (a=0; a<4; a++) A[a]=0; //azzero
        for (j=0; j<4; j++, k++)
            if (bin[k]=='1') A[j]=1; //k contatore cifre binarie processate
        val=8*A[0]+4*A[1]+2*A[2]+A[3]; //A[0]<<3+A[1]<<2+A[2]<<1+A[3]
        s[h++]=val + (val > 9 ? 'A'-10:'0'); //h contatore cifre esa
    }
    //terminazione stringa: 2 punti
    s[h]='\0'; //terminatore stringa
    return s;
}
```

### 3. (12 punti)

```
typedef struct l_tag {int value; struct l_tag *prev; struct l_tag *next;} list;

void ins_ordI(list **lpp, int value) {
    list *tmp_prev=NULL, *tmp_next;
    while (*lpp!=NULL && (*lpp)->value < value) {
        tmp_prev=*lpp; //salvo parte anteriore della lista
        lpp=&((*lpp)->next); //riposizionamento
    }
    //inserimento
    tmp_next=*lpp; //salvo parte posteriore (resto) della lista
    *lpp=(list *)malloc(sizeof(list));
    (*lpp)->value=value;
    (*lpp)->next=tmp_next; //collego nuovo a parte posteriore (resto) della lista
    (*lpp)->prev=tmp_prev; //collego nuovo a parte anteriore della lista
    if ((*lpp)->next != NULL)
        (*lpp)->next->prev=*lpp; //collego parte posteriore a nuovo
}
```