

Fondamenti di Informatica I

PROVA SCRITTA – 4 settembre 2001

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
- * La **correzione** di riferimento per l'autovalutazione verrà effettuata in questa stessa aula alle ore **11:45**
- * La consegna delle **fotocopie** dei compiti avverrà al termine della correzione
- * La **prova orale** si terrà **lunedì 10 settembre** 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. (10 punti)

Si scriva una funzione C che riceve in input un vettore di stringhe e che restituisce la stringa di 'valore' massimo (nel caso di più stringhe con stesso 'valore' massimo si restituisca la prima incontrata). Il 'valore' di una stringa sia definito come la somma dei codici ASCII dei caratteri costituenti la stringa stessa.

2. (6 punti)

Si scriva una funzione C che restituisce una lista dinamica (rappresentazione collegata mediante puntatore) di numeri interi, ordinata in modo crescente, dai valori dei nodi di un albero binario di ricerca.

3. (10 punti)

Si definisca una struttura del linguaggio C capace di rappresentare nome, cognome e numero di matricola di uno studente. Si scriva una funzione C che ordina un array di tali strutture in base a numeri di matricola crescenti utilizzando l'algoritmo quick-sort

4. (4 punti)

Si esprimano le seguenti dichiarazioni di variabile in linguaggio C:

- x e' un puntatore ad array di 10 variabili intere
- y e' un array di 10 array di 20 puntatori a double
- z e' un array di 10 puntatori ad array di 20 variabili intere

ESERCIZIO 1 (10 punti)

```
char *find_max_str_val(char *str_ptr[], int dim) {
    int i, j;
    int max_str_val=0, max_str_ind=0, tot;
    char c;
    for (i=0; i<dim; i++) {
        tot=0; j=0;
        while ((c=*(str_ptr[i]+j++))!=NULL)
            tot+=(int)c;
        if (tot > max_str_val) {
            max_str_val=tot;
            max_str_ind=i;
        }
    }
    if (max_str_val>0) return str_ptr[max_str_ind];
    else return NULL;
}
```

ESERCIZIO 2 (6 punti)

```
struct list {int val; struct list *next;};
struct btree {int val; struct btree *sx; struct btree *dx;};

void btree2list(struct btree *bt, struct list **l_ptr_ptr) {
    struct list *tmp;
    if (bt!=NULL) { //visita dal massimo al minimo
        btree2list(bt->dx,l_ptr_ptr);

        tmp=*l_ptr_ptr; //inserimento in testa
        *l_ptr_ptr=(struct list *)malloc(sizeof(struct list));
        (*l_ptr_ptr)->val=bt->val;
        (*l_ptr_ptr)->next=tmp;

        btree2list(bt->sx,l_ptr_ptr);
    }
}
```

ESERCIZIO 4 (4 punti)

int (*x)[10];

double *y[10][20];

int (*z[10])[20];

ESERCIZIO 3 (10 punti)

```
void swap(struct stud *a, int sx, int dx);
int partition(struct stud *a, int low, int high);
void quicksort(struct stud *a, int low, int high);

struct stud {
    char cognome[80];
    char nome[80];
    int mat;
};

void swap(struct stud *a, int sx, int dx)
{
    struct stud app;

    app=a[sx];
    a[sx]=a[dx];
    a[dx]=app;
}

int partition(struct stud *a, int low, int high )
{
    int sx, dx, pivot;
    int pivot_item;

    pivot = low;
    sx = low;
    dx = high;
    pivot_item = (a+low)->mat;
    while ( sx < dx ) {
        while(sx < dx  && (a+sx)->mat <= pivot_item ) sx++;
        while((a+dx)->mat > pivot_item ) dx--;
        if (sx < dx) swap(a, sx, dx);
    }
    // dx is final position for the pivot
    swap(a, low, dx);
    return dx;
}

void quicksort(struct stud *a, int low, int high )
{
    int pivot;
    if ( high > low ) {
        pivot = partition( a, low, high );
        quicksort( a, low, pivot-1 );
        quicksort( a, pivot+1, high );
    }
}
```