

## Fondamenti di Informatica I

PROVA SCRITTA – 12 luglio 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 **12:45**
- \* La consegna delle **fotocopie** dei compiti avverrà presso l'Istituto di Informatica al termine della correzione
- \* La **prova orale** si terrà **giovedì 19 luglio** alle ore **9:30** in aula **160/1**



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. (9 punti)

Dato un vettore di  $N$  elementi interi, si definisca una funzione  $C$  che crea dinamicamente e restituisce la matrice  $N \times 2$  in cui nella prima colonna vi sono gli elementi del vettore ordinati in modo crescente e nella seconda colonna gli indici che tali elementi avevano nel vettore originario (esempio:  $V[3]=\{15, 7, 22\} \rightarrow M[3][2]=\{\{7, 15, 22\}, \{1, 0, 2\}\}$ ).

### 2. (8 punti)

Si scriva una funzione  $C$  che inserisce un valore intero su un albero binario di numeri interi mantenendo l'albero bilanciato.

### 3. (8 punti)

Si definisca la funzione  $C$  che elimina da una lista di interi tutti gli elementi doppi (con lo stesso valore di un'altro elemento) rendendo disponibile al sistema la memoria da essi occupata.

### 4. (5 punti)

Si esprimano le seguenti dichiarazioni di funzioni (prototipi) in linguaggio C:

- A** è una funzione void con una matrice tridimensionale di interi come parametro
- B** è una funzione con un puntatore a un insieme di liste come parametro e ritorna un intero
- C** è una funzione che ritorna un array di interi.

### ESERCIZIO 1 (9 punti)

```
int * create_matrix(int *A,int size N);
void swap(int *M, int index1, index2);

int * create_matrix(int *A,int size N) {
    int *M=NULL, i, iter=0, null_iter=FALSE;

    if(N) {
        M = (int *)malloc(sizeof(int)*N*2);
        for(i=0; i < N; i++) {
            M[i*2] = A[i]; //metto vettore in prima colonna
            M[i*2+1] = i; //metto indice in seconda colonna
        }
        while((iter < N-1) && (null_iter == FALSE)) { //bubblesort
            null_iter=TRUE;
            for(i=0; i < N-1-iter; i++)
                if (M[i*2] > M[(i+1)*2]) {
                    swap(M, i*2, (i+1)*2); //swap element
                    swap(M, i*2+1, (i+1)*2+1); //swap index
                    null_iter=FALSE;
                }
            iter++;
        }
    } //endof if (N)
    return M;
}

void swap(int *M, int index1, index2) {
    int tmp;

    tmp=M[index1]; M[index1] = M[index2]; M[index2] = tmp;
}
```

### ESERCIZIO 2 (8 punti)

```
void weighted_insert(struct btree ** root_ptrptr, int value);
int get_num_nodes(struct btree * root_ptr);

void weighted_insert(struct btree ** root_ptrptr, int value) {
    int left_sum, right_sum;

    while(*root_ptrptr!=NULL) {
        left_sum = get_num_nodes((*root_ptrptr)->lx_ptr);
        right_sum = get_num_nodes((*root_ptrptr)->rx_ptr);
        if (left_sum >= right_sum)
            root_ptrptr = &((*root_ptrptr)->lx_ptr);
        else
            root_ptrptr = &((*root_ptrptr)->rx_ptr);
    }
    *root_ptrptr = (struct btree *)malloc(sizeof(struct btree));
    (*root_ptrptr)->lx_ptr = NULL;
    (*root_ptrptr)->rx_ptr = NULL;
    (*root_ptrptr)->value = value;
}

int get_num_nodes(struct btree * root_ptr) {
    if(root_ptr==NULL) return 0;
    else
        return (1 + get_num_nodes(root_ptr->lx_ptr) + get_num_nodes(root_ptr->rx_ptr));
}
```

ESERCIZIO 3 (8 punti)

```
void delete_copies(struct list * head_ptr);

struct list{
    int value;
    struct list * next_ptr;
};

void delete_copies(struct list * head_ptr) {
    struct list * moving_ptr, tmp_ptr;

    while (head_ptr != NULL) {
        moving_ptr = head_ptr;
        while (moving_ptr->next_ptr != NULL) {
            if (moving_ptr->next_ptr->value == head_ptr->value) {
                tmp_ptr = moving_ptr->next_ptr;
                moving_ptr->next_ptr = moving_ptr->next_ptr->next_ptr;
                free(tmp_ptr);
            }
            else moving_ptr = moving_ptr->next_ptr;
        }
        head_ptr = head_ptr->next_ptr;
    }
}
```

ESERCIZIO 4 (5 punti)

**void A ( int [T][R][ ]);** oppure: **void A (int \*, int T, int R, int C);**

**int B (struct list\_type (\*)[ ]);**

**impossibile far ritornare un array di interi**