

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

PROVA SCRITTA – 11 aprile 2005

### Avvertenze:

- \* Consegnare **solo fogli formato A4 scritti su un solo lato** (no fronte-retro)
- \* **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 **15:00**
- \* La consegna delle **fotocopie** dei compiti avverrà al termine della correzione
- \* La **prova orale** si terrà **giovedì 14 (N.O.) e giovedì 28 (F.C. e V.O.) aprile** alle ore **9:00** presso il D.I.I.G.A.



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. (12 punti)

Si definisca una funzione C che crea e restituisce la matrice di N righe e M colonne ottenuta dalla trasformazione della matrice di N righe e M colonne ricevuta in input in modo che ogni elemento della nuova matrice contenga la somma degli elementi da quello corrente fino all'ultimo della stessa riga, estremi compresi (esempio: la trasformazione di  $A[2][3]=\{\{11, 12, 13\}, \{21, 22, 23\}\}$  produce  $AT[2][3]=\{\{36, 25, 13\}, \{66, 45, 23\}\}$ ).

### 2. (12 punti)

Si definisca una funzione C che verifica se un albero binario (non di ricerca) contiene o meno due valori uguali.

### 3. (6 punti)

Se f1 e f2 sono definite nel seguente modo:

```
int f1(int *x) { int i=0,y=0;
                for (i=0; i<*x; i+=3) *x+=y++;
                return *x++; }
void f2(int *x, int **y) { *x+=f1(*y); }
```

Cosa stampa il seguente frammento di codice?

```
int x=3, y=5, *p=&y;
f2(&x, &p);
printf("\nx=%d, y=%d", x, (*p)++);
```

**Nota:** oltre al risultato riportare (magari graficamente) anche il ragionamento effettuato.

### 1. (12 punti)

Si definisca una funzione C che crea e restituisce la matrice di N righe e M colonne ottenuta dalla trasformazione della matrice di N righe e M colonne ricevuta in input in modo che ogni elemento della nuova matrice contenga la somma degli elementi da quello corrente fino all'ultimo della stessa riga, estremi compresi (esempio: la trasformazione di  $A[2][3]=\{\{11, 12, 13\}\{21, 22, 23\}\}$  produce  $AT[2][3]=\{\{36, 25, 13\}\{66, 45, 23\}\}$ ).

```
int *trasforma(int *A, int N, int M){
    int i, j, *AT=(int *)malloc(N*M);
    for (i=0; i<N; i++) {
        *(AT+i*M+M-1)= *(A+i*M+M-1); //ricopio l'ultimo della riga ...
        for (j=M-2; j>=0; j--) //... e parto dal penultimo all'indietro
            *(AT+i*M+j)= *(A+i*M+j)+*(AT+i*M+j+1);
    }
    return AT;
}
```

### 2. (12 punti)

Si definisca una funzione C che verifica se un albero binario (non di ricerca) di interi contiene o meno due interi uguali.

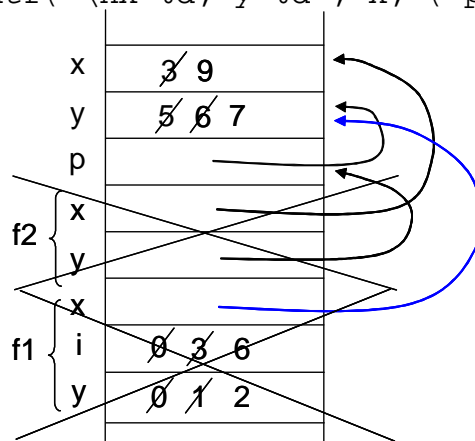
```
typedef struct btree_tag { int value; struct btree_tag *left;
                           struct btree_tag *right; } btree;

Boolean search(btree *root, btree *btp) { //btp è il nodo da cercare se doppio
    if (root==NULL) return FALSE;
    if (root->value==btp->value && root!=btp) return TRUE;
    return search(root->left, btp) || search(root->right, btp);
}

Boolean check_double(btree *root, btree *btp) {
    if (btp==NULL) return FALSE;
    if (search(root, btp)) return TRUE; //root non viene mai modificato qui
    return check_double(root, btp->left) || check_double(root, btp->right);
}
```

### 3. (6 punti)

```
int f1(int *x) { int i=0,y=0; for (i=0; i<*x; i+=3) *x+=y++;
                return *x++; }
void f2(int *x, int **y) { *x+=f1(*y); }
int x=3, y=5, *p=&y;
f2(&x, &p); printf("\nx=%d, y=%d", x, (*p)++);
```



x=9, y=6