

## Fondamenti di Informatica I

PROVA SCRITTA – 21 giugno 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 **martedì 3 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 + 5 = 14 punti)

Dato un vettore di N elementi interi e un intero P, si definisca una funzione C che:

- a) senza effettuare l'ordinamento degli elementi del vettore e senza utilizzare dei vettori d'appoggio, scambia le posizioni di alcuni elementi affinché tutti gli elementi del vettore con valori inferiori o uguali a P (primo insieme) si trovino a sinistra degli elementi con valori maggiori di P (secondo insieme);
- b) restituisce 2 alberi binari ordinati (alberi binari di ricerca) di cui uno costruito con tutti e soli gli elementi del primo insieme e l'altro con quelli del secondo insieme.

### 2. (8 punti)

Dati i 2 alberi binari di ricerca dell'esercizio precedente (per i quali vale la proprietà che il valore associato a un qualsiasi nodo del primo albero è inferiore al valore di un qualsiasi nodo del secondo albero), si definisca la funzione C che riceve in ingresso i 2 alberi binari di ricerca e restituisce l'albero binario di ricerca che scaturisce dalla loro fusione.

### 3. (8 punti)

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

- A** e' un array di 3 puntatori a puntatori a interi
- B** e' un array di 50 record (strutture) composti da un intero e 2 array di 25 caratteri
- C** è un puntatore a un array di 3 puntatori a interi
- D** è una funzione void con un intero e 2 puntatori a float come parametri
- E** è una funzione che ritorna un puntatore a caratteri e ha un intero come parametro
- F** e' un array di puntatori a funzioni che ritornano un intero

```

void swap_element(int *V, int sx, int dx)
{ int app; app=V[sx]; V[sx]=V[dx]; V[dx]=app; }

void arr2btree(int *V, int N, int P,
               struct btree **root1_ptr_ptr, struct btree **root2_ptr_ptr)
{ int sx=0, dx=N-1, i;
  while (sx < dx) { //array partition (similar to step 1 of quicksort)
    while ((dx > 0) && (V[dx] > P)) dx--;
    while ((sx < N-1) && (V[sx] <= P)) sx++;
    if (sx < dx) swap_element(V, sx, dx);
  }
  if (sx==dx) { //one set is empty
    if (V[dx] > P) dx=-1; //all elements > P (first set is empty)
    else dx=N-1; //all elements <= P (second set is empty)
  } //dx = last element (max index) of left partition
  //creazione 2 btree
  for (i=0; i<=dx; i++) recursive_ordered_insertion(root1_ptr_ptr, V[i]);
  for (i=dx+1; i<=N-1; i++) recursive_ordered_insertion(root2_ptr_ptr, V[i]);
}

void rec_ordered_insertion(struct btree **root_ptr_ptr, int val)
{ if((*root_ptr_ptr)!=NULL) {
  if((*root_ptr_ptr)->value > val)
    recursive_ordered_insertion(&((*root_ptr_ptr)->left_ptr), val);
  else
    recursive_ordered_insertion(&((*root_ptr_ptr)->right_ptr), val);
} else {
  (*root_ptr_ptr)=(struct btree *)malloc(sizeof(struct btree));
  (*root_ptr_ptr)->value=val;
  (*root_ptr_ptr)->left_ptr=NULL;
  (*root_ptr_ptr)->right_ptr=NULL;
}
}

void btree_merge(struct btree **root1_ptr_ptr, struct btree *root2_ptr)
{ if ((*root1_ptr_ptr)==NULL) *root1_ptr_ptr=root2_ptr;
  else if (root2_ptr!=NULL) { //entrambi gli alberi non sono NULL
    do {
      if( (*root1_ptr_ptr)->value > root2_ptr->value)
        root1_ptr_ptr=&((*root1_ptr_ptr)->left_ptr);
      else
        root1_ptr_ptr=&((*root1_ptr_ptr)->right_ptr);
    } while((*root1_ptr_ptr)!=NULL);
    *root1_ptr_ptr=root2_ptr;
  }
}

```

```

int ** A [3];
struct { int a; char b[25]; char c[25]; } B [50];
int (* C) [3];

```

```

void D (int, float **, float **);
char * E ( int );
int (* F [ ] ) ();

```