

Fondamenti di Informatica I

PROVA SCRITTA – 30 ottobre 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 **10:45**
- * La consegna delle **fotocopie** dei compiti avverrà al termine della correzione
- * La **prova orale** si terrà **martedì 6 novembre** alle ore **9:00** presso l'**Istituto di Informatica**



Si ricorda che chi si presenterà all'orale **DEVE evidenziare sulle fotocopie dei compiti, con penna rossa o blu, gli errori commessi** e portare l'implementazione al computer della propria soluzione, eventualmente corretta, **corredata di tutto quanto necessario alla verifica** del corretto funzionamento.

1. (10 punti)

Si consideri un albero n-ario rappresentato su un albero binario secondo la convenzione per cui il figlio primogenito va sul sotto-albero sinistro del padre ed eventuali fratelli vanno sul sotto-albero destro del fratello immediatamente maggiore. A ciascun nodo siano associati un'etichetta alfanumerica e un valore intero.

Si definisca la struttura dati in linguaggio C e si definisca la funzione C che ricerca nell'albero tutti i nodi con una etichetta assegnata e che ritorna la somma dei valori interi di tutti i figli di tali nodi.

Suggerimento: la seguente funzione “*int strcmp(const char *s1, const char *s2);*” ritorna 0 solo se le stringhe s1 e s2 sono uguali.

2. (20 punti)

Sia data la seguente struttura dati in C (record):

```
struct articolo {int codice; int pezzi; double prezzo}
```

Si scriva una funzione C che:

- legge da un file di testo dal nome assegnato tutti (numero sconosciuto) i record “articolo” presenti (uno per riga) (3 punti),
- raggruppa tutti i record con lo stesso “codice” in un unico record in cui “pezzi” vale la somma dei “pezzi” di ciascun record e “prezzo” vale il prezzo medio; quindi, scrive i record risultanti ordinati in base al “codice” in un altro file di testo dal nome assegnato (14 punti),
- restituisce e scrive in fondo a quest'ultimo file la valorizzazione dell'inventario data dalla somma dei prodotti “pezzi * prezzo” di ciascun record.scritto (3 punti).

Si consideri un albero n-ario rappresentato su un albero binario secondo la convenzione per cui il figlio primogenito va sul sotto-albero sinistro del padre ed eventuali fratelli vanno sul sotto-albero destro del fratello immediatamente maggiore.

```
# include <stdio.h>
# include <stdlib.h>
#include <string.h>

struct btree {
    char id[81];
    int value;
    struct btree * lx;
    struct btree * rx;
};

void visit_tree(struct btree *root, char *id, int *sum);

void visit_tree(struct btree *root, char *id, int *sum)
{
    if (root != NULL) {
        visit_tree(root->lx, id, sum);
        visit_tree(root->rx, id, sum);
        if (strcmp(root->id,id)== 0)
            if (root->lx != NULL) { //esiste primo figlio
                (*sum) += root->lx->value;
                root = root->lx->rx; //vado su secondo figlio
                while (root != NULL) { //esistono altri fratelli
                    (*sum) += root->value;
                    root = root->rx;
                }
            }
    }
}
```

```

struct articolo {int codice; int pezzi; double prezzo;};
struct list {struct articolo elem; struct list *next;};
long inventario(char *file_in, char *file_out);
void insert(struct list **root, struct articolo riga);

long inventario(char *file_in, char *file_out) {
    FILE *in, *out;
    struct articolo riga;
    struct list *root=NULL;
    double tot=0.0;

    if ((in=fopen(file_in, "rt"))!=NULL) { //OK apertura file input
        while (!feof(in)) {
            fscanf(in,"%d %d %lf\n", &riga.codice, &riga.pezzi, &riga.prezzo);
            insert(&root, riga); //inserimento ordinato in una lista
        }
        fclose(in);

        if ((out=fopen(file_out, "wt"))!=NULL) { //OK apertura file output
            while (root != NULL) {
                fprintf(out, "%d %d %lf\n", root->elem.codice,
                    root->elem.pezzi, root->elem.prezzo);
                tot += root->elem.pezzi * root->elem.prezzo;
                root = root->next;
            }
            fprintf(out, "%lf\n", tot);
            fclose(out);
        }
    }
    return tot;
}

void insert(struct list **root, struct articolo riga)
{ //inserimento ordinato (in base al codice articolo) in una lista
    int found=0;
    struct list *tmp, **curr=root;

    while (!found && (*curr)!=NULL) {
        if ((*curr)->elem.codice == riga.codice) {
            //codice già esistente: aggiorno numero pezzi e nuovo prezzo medio
            (*curr)->elem.prezzo = ((*curr)->elem.pezzi * (*curr)->elem.prezzo +
                riga.pezzi * riga.prezzo) /
                ((*curr)->elem.pezzi + riga.pezzi);
            (*curr)->elem.pezzi += riga.pezzi;
            found=1; //terminazione
        }
        else if ((*curr)->elem.codice > riga.codice) {
            tmp=(*curr); //devo inserire un nuovo articolo
            (*curr)=(struct list *)malloc(sizeof(struct list));
            (*curr)->elem = riga; //uguaglianza tra strutture
            (*curr)->next=tmp;
            found=1; //terminazione
        }
        else curr=&((*curr)->next); //continuo ricerca
    } //end of while
    if (!found) {
        (*curr)=(struct list *)malloc(sizeof(struct list));
        (*curr)->elem = riga; //uguaglianza tra strutture
        (*curr)->next=NULL;
    }
}

int main() {long tot; tot=inventario("c:\\aaa.txt", "c:\\aab.txt");}

```