Corso di Laurea in Ingegneria Informatica (NOD)
Fondamenti di Informatica B
Prof. Maurizio Vincini

Prova Scritta - 16/06/2016

ESERCIZIO 1 (7 punti)
Dato il seguente listato:

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

int calc1(int a, int b, int c) {
int d;

for(d=0;d<2;d=d+2)
    a=a+b;c=b-a;

printf("valori a: %d, b:%d, c:%d, d : %d\n", a, b, c, d);
return b; }

int funz1 (int x, int *y) {
int z = 2;


z = calc1(x + 1, *y, z);
x++;
*y = x + z;
z = *y - x;

printf("valori x: %d, y:%d, z:%d\n", x, *y, z);

return z; }


void main() {
int a = 6, b = 3, c = 3, i;
for(i=0;i<2;i=i+2)
    c += funz1(a,&b);
printf("scrivi a:%d, b:%d, c:%d \n", a, b, c);
return; }
```

Scrivere la corretta sequenza della stampa a video e mostrare lo sviluppo dei record di attivazione ignorando le chiamate alle funzioni di I/O.

ESERCIZIO 2 (9 punti)

Abbiamo a disposizione N cavi elettrici dotati agli estremi di connettori che possono essere di tipo 0, 1, 2 (uno di ingresso ed uno di uscita).
Il singolo cavo può essere rappresentato dalla seguente struttura dati:

```
typedef struct  {
        int     input;
        int     output;
} cavo;
```

Trovare, se esistono, le sequenze in cui disporre tutti gli N cavi che permettono la connessione completa, cioè nelle quali ogni cavo intermedio possiede un connettore di input dello stesso tipo di quello di output del suo predecessore e di output identico a quello di input del successore.

ESERCIZIO 3 (8 punti)

Dato un file di testo “magazzino.txt” che contiene su ciascuna riga il nome di un articolo e il suo prezzo, creare in memoria una lista dinamica che contenga gli articoli ordinati rispetto al prezzo (si suppone che i nomi non siano ripetuti).

Visualizzare, usando la lista, per ciascun prezzo, il numero di articoli presenti.

ESERCIZIO 4 (9 punti)

Dati due alberi binari BST contenenti valori interi, scrivere una funzione che torna 1 se gli alberi contengono gli stessi elementi, 0 altrimenti.

ESERCIZIO 3 (8 punti)

Dato un file di testo “magazzino.txt” che contiene su ciascuna riga il nome di un articolo e il suo prezzo, creare in memoria una lista dinamica che contenga gli articoli ordinati rispetto al prezzo (si suppone che i nomi non siano ripetuti).

Visualizzare, usando la lista, per ciascun prezzo, il numero di articoli presenti.

```
typedef char stringa[20];
typedef struct {
     stringa nomeArticolo;
     int prezzo;

} element; /* DEFINIZIONE */


typedef int boolean; /* DEFINIZIONE */

boolean isEqual(element e1,element e2) {
return (e1.prezzo == e2.prezzo);
}
boolean isLess(element e1, element e2) {
return (e1.prezzo < e2.prezzo);
}

void main(void) {
     FILE *file;
     char nomeArticolo[20];
     int prezzo;
     element a;
     list lista = emptylist();
     int i = 0;

     file = fopen("../magazzino.txt", "r");

     if (file == NULL) {
     printf("Errore Apertura file");
     return;
     }


     while (!feof(file))
     {
          fscanf(file, "%s", a.nomeArticolo);
          fscanf(file, "%d", &a.prezzo);


          lista = insord(a, lista);
          i++;
     }

     showlist(lista);
     contaArticoli(lista);
}
```

```
list insord(element e, list l){
     list l1 = NULL, root = l;
     list t;
     if(empty(l) || !isLess(head(l), e)){
          t = cons(e, l);
     return t;
     }

     t = cons(e, NULL);
     while(!empty(l) && isLess(head(l), e)){
          l1=l;
          l = tail(l);
     }

     l1->next = t;
     t->next = l;

     return root;
}

void contaArticoli(list l) {

     list root = l;
     int prezzo, cont;


     while (!empty(l)){
          prezzo = head(l).prezzo;
          cont=0;

          while (!empty(l) && prezzo == head(l).prezzo){
               cont++;
               l=tail(l);
               }
          printf("\nArticoli presenti con prezzo %d: %d", prezzo,
cont);

          //l=tail(l);
     }

     return;
}
```

ESERCIZIO 4 (9 punti)

Dati due alberi binari BST contenenti valori interi, scrivere una funzione che torna 1 se gli alberi contengono gli stessi elementi, 0 altrimenti.

```
void main () {
     tree t1,t2;
     FILE *f;
     list l = NULL;
     element el;
     int ris = -1;

     f = fopen("../Valori3.txt", "r");
     if (f == NULL) {
          printf("Errore Apertura file");
          return;
     }
     t1 = emptyTree();

     while (fscanf(f, "%d", &el)>0) {
          t1 = insertBinOrd(el, t1);
     }

     fclose(f);

     f = fopen("../Valori4.txt", "r");
     if (f == NULL) {
          printf("Errore Apertura file");
          return;
     }
     t2 = emptyTree();

     while (fscanf(f, "%d", &el)>0) {
          t2 = insertBinOrd(el, t2);
     }

     fclose(f);

  printf("\nVisita in ordine t1\n");
  inOrder(t1);
  printf("\nVisita in ordine t2\n");
  inOrder(t2);

  if (conta(t1) == conta(t2))
       ris = confronta(t1,t2);
  else
       ris = 0;

  printf("\nRisultato controllo: %d", ris);

  printf("\nElementi di t1: %d", conta(t1));
  printf("\nElementi di t2: %d", conta(t2));
}
```

```
int conta(tree t1) {

  if (!empty(t1)) {

     return 1 + conta(left(t1))  + conta(right(t1));
  }
  else
     return 0;
}

int cerca(tree t, element e) {

     if (!empty(t)) {

          if (root(t) == e)
               return 1;

          else

               if (e < root(t))
                    return cerca(left(t),e);
               else
                    return cerca(right(t),e);

     }
     else
          return 0;

}

int confronta(tree t1, tree t2) {
     int ris=-1;
  if (!empty(t1)) {
     if (confronta(left(t1), t2) == 0)
          return 0;

     printf("\nCerco %d in t2",root(t1));
     printf("\nRitorno %d",cerca(t2,root(t1)));
    if (cerca(t2,root(t1)) == 0)
          return 0;
     if (confronta(right(t1), t2) == 0)
          return 0;
  }

  return 1;
}
```