Informatica III – 23/4/13 – info 3

Al PC

1.Record di attivazione A

Scrivi una funzione in **C** che dato in ingresso una stringa (come array di char terminata da 0) restituisce il numero di vocali in essa (definisci una funzione isVocale(char)). Al solito, scrivi tre versioni: una non ricorsiva, una ricorsiva senza tail recursion e una ricorsiva con tail recursion.

Specifica esattamente i parametri che passi alla procedura, il tipo di passaggio utilizzato e il loro significato.

Scrivi anche un main di esempio in cui chiami la funzione con la string "ab" invocala in modo assegni il risultato ad una variabile globale VOCALI. Non usare alcuna altra variabile globale.

Disegna il record di attivazione per tutte e tre le versioni fino alla massima estensione del record di attivazione. Nel caso di tail recursion, spiega quali ottimizzazioni hai adottato o potresti adottare.

Chiama il file cognome_nome_es1.c

2. Dangling pointer sullo stack

Scrivi un esempio di una funzione in cui hai un dangling pointer sullo stack. Scrivi un main in cui mostri un caso in cui il difetto non causa alcun malfunzionamento e un esempio in cui il difetto causa un malfunzionamento. Spiega con commenti nel codice.

Chiama il file cognome_nome_es2.cpp

3. Java Visitor Pattern

Implementa un vistor pattern in Java in cui devi visitare una gerarchia di Figure (Figura, Rettangolo e Cerchio) con un calcolo dell'area. Ricorda che le Figure dovranno essere Visitable, mentre l'operazione dovrà essere un Visitor. Puoi anche non implementare del tutto i metodi di calcolo. Scrivi anche un main di esempio in cui crei alcune figure e calcoli l'area.

Fai il tutto in un progetto con tuo cognome e nome.

4.Key-Hoare

Scrivi un programma che effettua il prodotto mediante somme successive in key hoare, scrivi pre (meglio se pre = true) e post condizioni e dimostra la correttezza. Guarda il file di esempio nello zip fornito.

SU FOGLIO

C++

Considera il seguente codice?

```
#include <iostream>
using namespace std;
class veicolo {
private:
      int pri() {
            return 1;
      }
public:
      int pub() {
            return 2;
      }

      virtual int vpub() {
            return 3;
      }
};

class camper: private veicolo {
public:
      int vpub() {
            return 5;
      }
};

class automobile: public veicolo {
private:
      int pri() {
            return 6;
      }
public:
      int vpub() {
            return 7;
      }
};
 void f_veicolo(veicolo v) {
   cout << v.pub() << " ";
   cout << v.pri() << " ";
       cout << v.vpub() << endl;
 }
 void f_camper(camper c) {
   cout << c.pub() << " ";
   cout << c.pri() << " ";
   cout << c.vpub() << endl;
 }
 void f_automobile(automobile a) {
   cout << a.pub() << " ";
   cout << a.pri() << " ";
   cout << a.vpub() << endl;
 }
 void f_p_veicolo(veicolo* v) {
   cout << v->pub() << " ";
   cout << v->pri() << " ";
   cout << v->vpub() << endl;
 }
 void f_p_camper(camper* c) {
   cout << c->pub() << " ";
   cout << c->pri() << " ";
   cout << c->vpub() << endl;
 }
 void f_p_automobile(automobile* a) {
   cout << a->pub() << " ";
   cout << a->pri() << " ";
   cout << a->vpub() << endl;
 }
 void f_r_veicolo(veicolo& v) {
   cout << v.pub() << " ";
   cout << v.pri() << " ";
   cout << v.vpub() << endl;
 }
 void f_r_camper(camper& c) {
   cout << c.pub() << " ";
   cout << c.pri() << " ";
   cout << c.vpub() << endl;
 }
 void f_r_automobile(automobile& a) {
   cout << a.pub() << " ";
   cout << a.pri() << " ";
   cout << a.vpub() << endl;
 }
 int main() {
      veicolo v;
      camper c;
      automobile a;
   //
   1. f_veicolo(v);
   2. f_veicolo(c);
   3. f_veicolo(a);
   //
   4. f_p_veicolo(&v);
   5. f_p_veicolo(&c);
   6. f_p_veicolo(&a);
   //
   7. f_r_veicolo(v);
   8. f_r_veicolo(c);
   9. f_r_veicolo(a);
 }
```

Quale è l'output/effetto prodotto da ogni linea numerata del main? Se una funzione contiene un errore spiega l'errore e ignora la linea (ed eventuali istruzioni che dipendono da essa).

Dinamic Binding in Java

Date le seguenti dichiarazioni:

```
class A{
   public int m(long d){return 1;}
   public int f(double d){return 2;}
}
```

```
class B  extends A{
   public int m(short d){return 3;}
   public int f(double d){return 4;}
}
```

```
A ab = new B();
A aa = new A();
```

Quale è il valore delle seguenti tre espressioni spiegando bene (cioè anche il processo di early e late binding dove necessario) il perché (anche se le ritieni errate) – ricorda che un numero decimale è di default double in Java.

```
1.aa.m(2);
2.aa.m((short)2);
3.aa.m(2.0);
4.aa.f(2.5f);// EB: cerca in A un f(float). Trovo f(double) e lo seleziono
            //LB: cerco in A un f(double), lo trovo, eseguo => 2
5.ab.f(2.5);// EB: cerco in A un f(double). Lo trovo e lo seleziono
            // LB: cerco in B un f(double). Lo trovo e lo eseguo => 4
6.ab.m(2); // EB: cerco in A un m(int). Trovo m(long), va bene, e lo seleziono
           // LB: cerco in B un m(long). Non trovo in B, salgo in A, trovo ed
eseguo, => 1
7.ab.m((short)2);// EB: cerco in A un m(short). Trovo m(long), va bene, e lo
seleziono
           // LB: cerco in B un m(long), non lo trovo, in A sì, => 1
8.ab.m(2.0); //EB: cerco in A un m(double), non trovo ERRORE COMPILAZIONE
9.  ab.f(2.5f);//EB: cerco in A un f(float), ok, trovo f(double), lo seleziono
// LB, cerco in B un f(double), lo trovo, eseguo =>  4
10. ab.f(2.5);// idem (cerco però direattemnte un f(double)) => 4
```

Testing – 6 crediti

Testing programmi

Dato il seguente metodo che dice se uno può parteciare ad una festa:

```
public boolean allowed(int eta, char gender) {
  if ( (eta >= 20 && gender == 'M') || (eta >=18 && gender ='F'))
      return true;
    else
      return false;
}
```

Trova i casi di test per la copertura delle istruzioni, delle decisioni, delle condizioni e l'MCDC

Testing FSM

Scrivi una macchina a stati finiti con almeno 3 stati e trovane un transition tuor **euleriano**. Introduci un difetto e scopri se il tuo test è in grado di scoprirlo.