1. Record di attivazione A

Scrivi una funzione in **C** che dato in ingresso un array di interi restituisce il massimo nell'array se l'array non è vuoto, altrimenti restituisce MIN_INT (che è il valore più piccolo da assegnare ad un int). 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 un array di esempio e invocala in modo assegni il risultato ad una variabile globale. 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.

2. Recordi di attivazione in C++

Considera questo codice C++:

```
void foo1(int &y){
          cout << "y = " << y << endl;
          y = 6;
          cout << "y = " << y << endl;
}
int foo2(int y){
          int x = 20;
          cout << "y = " << y << endl;
          y = 6;
          cout << "y = " << y << endl;
          return y;
}
int main(){
          int x = 5;
          cout << "x = " << x << endl;
          foo1(x);
          cout << "x = " << x << endl;
          x = foo2(x);
          cout << "x = " << x << endl;
          return 0;
}
```

Quale è l'output? Scrivi il record di attivazione per la chiamata delle due funzioni.

3. Cyclone

A cosa servono i puntatori never-`NULL`? Come si usano? Fai un piccolo esempio.

4. Dangling pointer in C++

Si possono avere dangling pointer anche in C++? Come? Come si possono evitare e con quali conseguenze (sia positive che negative)? Fai un esempio.

5. C++

Considera il seguente codice?

```
class Edificio {
 private:
   int age() { return 1; }
 public:
   int color() { return 2; }
   virtual int price() { return 3; }
};

class Villa: private Edificio {
 public:
   int price() { return 5; }
};

class Magazzino: public Edificio {
 private:
  int age() { return 6; }
 public:
  int price() { return 7; }
};
int main() {
 1.    Edificio p;
 2.    cout << p.age() << endl;
 3.    cout << p.color()<< endl;

 4.    Villa s;
 5.    cout << s.age() << endl;
 6.    cout << s.color()<< endl;
 7.    cout << s.price()<< endl;

 8.    Magazzino m;
 9.    cout << m.age() << endl;
 10.   cout << m.color()<< endl;
 11.   cout << m.price()<< endl;

 12.   Villa * pv = &p;
 13.   Magazzino * pm = &p;

 14.   Edificio * pp = &s;
 15.   cout << pp->price()<< endl;
 16.   cout << pp->color()<< endl;

 17.   Edificio * pp2 = &m;
 18.   cout << pp2->price()<< endl;
 19.   cout << pp2->color()<< endl;

 20.   Edificio e1 = s;
 21.   cout << e1.age() << endl;
 22.   cout << e1.color()<< endl;
 23.   cout << e1.price()<< endl;

 24.   Edificio e2 = m;
 25.   cout << e2.age() << endl;
 26.   cout << e2.color()<< endl;
 27.   cout << e2.price()<< endl;}
```

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

6. Dinamic Binding in Java

Date le seguenti dichiarazioni:

```
class Edificio{

   public int m(float d){return 1;}

}


Object ov = new Edificio();

Edificio e = new Villa();
```

```
class Villa  extends Edificio{

   public int m(float d){return 2;}

   public int m(int d){return 3;}

}

Villa v = new Villa();
```

Quale è il valore delle seguenti espressioni spiegando bene (cioè anche il processo di early e late binding dove necessario) il perché (anche se le ritieni errate):

```
ov.equals(ov);

e.m(2);

e.m(2.0f);
```

```
v.m(2);

v.m(2.0);
```

7. Java

Dato il seguente metodo:

```
public void findMax(Collection<Edificio> l) {…}
```

Assumi che Villa estende Edificio. Vector implementa Collection.

Quali di queste chiamate sono corrette? Se non lo sono perché e come potrei modificare la segnatura del metodo per renderle corrette (se possibile)?

```
1. findMax(new Vector<Edificio>());

2. findMax(new Vector<Villa>());

3. findMax(new Collection<Edificio>());
```

8. Semantica assiomatica

Scrivi un piccolo programma che calcola il prodotto di x per y usando solo somme e sottrazioni, scrivi le precodizioni e le postcondizioni e cerca di dimostrare la correttezza.

9. Testing [6 crediti]

Dato il seguente programma:

```
foo(int x, int y, int z){
     if (x > 2){
        if ((y <x && y >z) || ( y ==5 && z == y+1)){
                z++;
        } else{
                z--;
        }
    }
}
```

Disegna il grafo del programma e trova i casi di test per la copertura delle istruzioni, delle decisioni, delle condizioni e l'MCDC

SOLUZIONI

```
Dinymic Binding

v.m(2):

EB cerca in Villa un metodo m(int), candidati ce ne sono due m(int) e m(float),
quello selezionato è m(int) che non richiede promozioni

LB: v è ref oggetto della classe Villa, cerca in Villa m(int), lo trova e lo esegue,
return 3

v.m(2.0);

EB: cerca in Villa un metodo m(double), non c'è neanche uno adatto: errore di
compilazione
```