1. Record di attivazione

Scrivi una funzione in **C** che dato in ingresso un array di int restituisce il numero minore. 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. Descrivi anche le assunzioni che fai (ad esempio zero terminated o cose simili).

Scrivi anche un main di esempio in cui chiami le funzioni con un array di tua scelta con 2 elementi. L'array deve essere dichiarato come variabile nel main. La variabile **MIN** a cui assegnare nel main il risultato della funzione di cui sopra deve essere globale. Non usare altre variabili globali.

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. Cyclone

Riscrivi il metodo non ricorsivo nell'esercizio 1 usando Cyclone. Valuta diverse alternative nel passaggio dei parametri.

3. Riferimenti in C++

Scrivi una piccola funzione che prende un parametro come riferimento e un frammento di codice che la chiama. Quali sono i vantaggi/svantaggi del passaggio per riferimento rispetto al passaggio per valore?

4. C++

Considera il seguente codice?

```
class Figura {
public:
  int draw() { return 1; }
  int color() { return 2; }
  virtual int move() { return 3; }
};

class Square : private Figura {
public:
  int draw() { return 5; }
  int move() { return 7; }
};

class Rectangle : public Figura {
public:
  int draw() { return 8; }
  int move() { return 9; }
};
```

```
int main() {
  Square quad; // ok
  cout << quad.draw() << endl; // 5
  cout << quad.color()<< endl; // errore è privato in square
  cout << quad.move()<< endl; // 7
  Figura* p = &quad;  // errore derivazione privato !!!
```

```
  cout << p->move()<< endl; //skip
  Rectangle r; //ok
  Figura p2 = r; // OK (non c'è polimorfismo però)
  cout << p2.draw() << endl; // draw di figura: 1
  cout << p2.color()<< endl; // idem 2
  cout << p2.move()<< endl;  // 3
}
```

Quale è l'output/effetto prodotto da ogni linea del main? Se contiene un errore scrivi errore, spiega l'errore e ignora la linea.

5. Dinamic Binding in Java

Date le seguenti dichiarazioni:

```
class Computer{

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

   public int f(int d){return 2;}

}
```

```
class Desktop extends Computer{

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

   public int f(int d){return 4;}

}
```

```
Object od = new Desktop();

Computer cd = new Desktop();
```

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):

```
od.m(1);

cd.m(1);

cd.m(1L);

cd.f(1);
```

6. Java

Ereditarietà e Sottotipazione in Java con un esempio.

7. Semantica assiomatica

Considera il seguente programma che dovrebbe calcolare in d la differenza di x e y:

```
d = x;
while (y != 0) {
    d--;
    y--;
}
```

Come scriveresti le precondizioni e postcondizioni? Come dimostreresti la sua correttezza?