# Lab Sheet 3: Recursive Algorithms in C

## Title: Recursive Algorithms – Tail and Non-Tail Method (Factorial, Sum of Natural Numbers, Fibonacci, Tower of Hanoi)

**Objective:**

- Understand recursion in C.
- Differentiate and implement Tail Recursion vs Non-Tail Recursion.
- Practice recursive approaches for classical problems.

## Part A: Recursive Function Implementations

### 1. Factorial

**Non-Tail Recursive:**

```
int factorial_non_tail(int n) {
    if (n == 0)
        return 1;
    return n * factorial_non_tail(n - 1);
}
```

**Tail Recursive:**

```
int factorial_tail(int n, int acc) {
    if (n == 0)
        return acc;
    return factorial_tail(n - 1, acc * n);
}
```

### 2. Sum of Natural Numbers

**Non-Tail Recursive:**

```
int sum_non_tail(int n) {
    if (n == 0)
        return 0;
    return n + sum_non_tail(n - 1);
}
```

**Tail Recursive:**

```
int sum_tail(int n, int acc) {
    if (n == 0)
        return acc;
    return sum_tail(n - 1, acc + n);
}
```

### 3. Fibonacci Series

**Non-Tail Recursive:**

```
int fib_non_tail(int n) {
    if (n <= 1)
        return n;
    return fib_non_tail(n - 1) + fib_non_tail(n - 2);
}
```

**Tail Recursive:**

```
int fib_tail(int a, int b, int n) {
    if (n == 0)
        return a;
    return fib_tail(b, a + b, n - 1);
}
```

**Part B: Tower of Hanoi**

```
void tower_of_hanoi(int n, char source, char target, char aux) {
  if (n == 1) {
    printf("Move disk 1 from %c to %c\n", source, target);
    return;
  }
  tower_of_hanoi(n - 1, source, aux, target);
  printf("Move disk %d from %c to %c\n", n, source, target);
  tower_of_hanoi(n - 1, aux, target, source);
}
```

**Lab Tasks:**

1. Write and test each function for:

   Factorial (Tail and Non-Tail)

   Sum of Natural Numbers (Tail and Non-Tail)

   Fibonacci (Tail and Non-Tail)

   Tower of Hanoi (n = 3 disks)

2. Modify main() to demonstrate each of the above implementations.
3. Print outputs and trace recursive calls where applicable.
4. Challenge: Convert any other recursive algorithm (e.g., GCD, Reverse Array) to tail-recursive form.