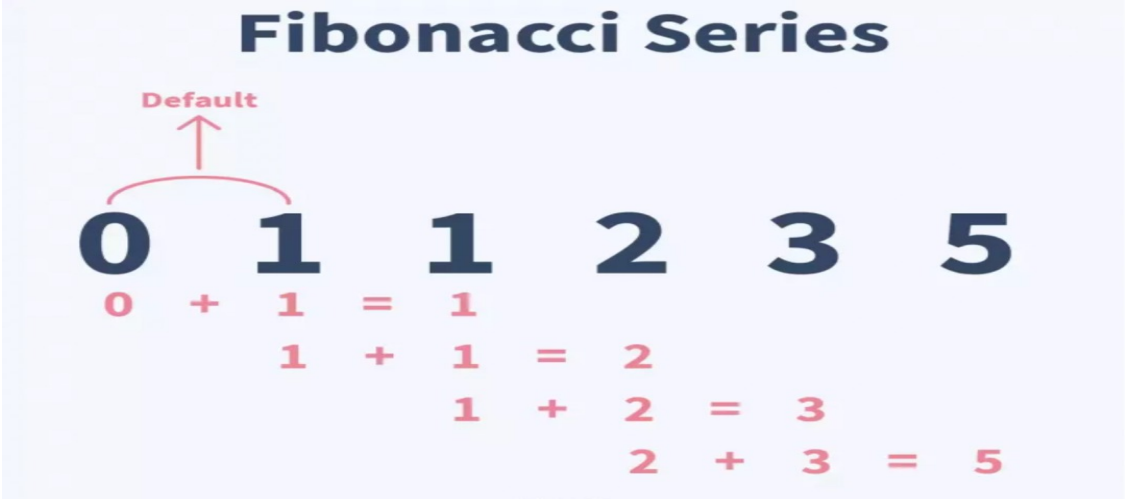
LAB 3

Introduction:

Recursion is a powerful programming technique where a function calls itself in order to solve a problem.. In this report, we delve into the implementation of recursion in C, through two important algorithms: Fibonacci Series and Tower of Hanoi.

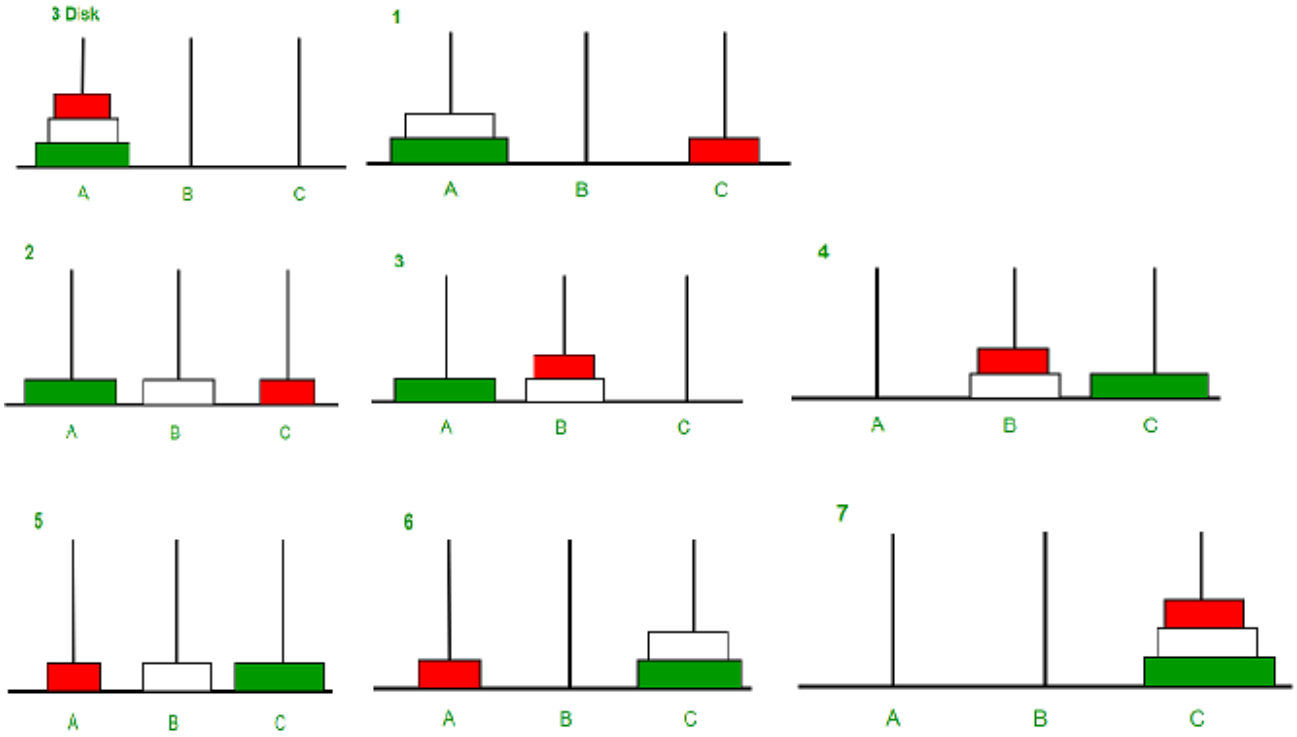
Fibonacci Series:
The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding ones. The sequence starts with 0 and 1. Mathematically, it is defined by the recurrence relation F(n) = F(n-1) + F(n-2), with initial conditions F(0) = 0 and F(1) = 1.



In this C program, we'll explore how to generate Fibonacci series using recursion.
The recursive approach for the Fibonacci series involves defining a function that calculates the nth Fibonacci number by calling itself for the (n-1)th and (n-2)th Fibonacci numbers.

Tower of Hanoi:
The Tower of Hanoi is a classic problem in computer science and mathematics. It involves three pegs and a number of disks of different sizes, which can slide onto any peg. The puzzle starts with the disks in a neat stack in ascending order of size on one peg, the smallest at the top, thus making a conical shape.

The objective of the puzzle is to move the entire stack to another peg, obeying the following simple rules:



1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty peg.
3. No disk may be placed on top of a smaller disk.

The Tower of Hanoi problem is often used to illustrate the principles of recursion. The recursive solution involves breaking down the problem into smaller, more manageable subproblems.
In this C program, we'll implement the Tower of Hanoi using recursion.

Algorithms

1. Fibonacci Series:

- If n is 0 or 1, return n (base case).
- Otherwise, calculate F(n) by recursively calling fibonacci(n - 1) and fibonacci(n - 2), then sum the results.
- The base case ensures that the recursion stops when n is 0 or 1.

2. **Tower Of Hanoi**

- If n is 0, do nothing (base case).
- Otherwise, move n-1 disks from the source peg to the auxiliary peg using the target peg as an auxiliary peg. Move the nth disk from the source peg to the target peg.
- Finally, move the n-1 disks from the auxiliary peg to the target peg using the source peg as an auxiliary peg.

DISCUSSION

Though this lab we learned the implementation of linked list in c programming language. Although we had studied the algorithm already, actually implementing it with the proper syntax and error conditions proved to be bit of a challenge. The traverse algorithm was took a bit of carefulness in loop design. After few tries and carefullness, the program worked sucessfully. The lab was completed as a sucessful learning experience.

CONCLUSION

We hereby conclude the lab report on linked list implementation in c programming language.

CODE:

Fibonacci series

```
#include <stdio.h>

int fibonacci(int n) {
    if (n <= 1) {
        return n;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}

void generateFibonacci(int n) {
    printf("Fibonacci Series up to %d terms: ", n);

    for (int i = 0; i < n; i++) {
        printf("%d, ", fibonacci(i));
    }

    printf("\n");
}

int main() {
    int terms;

    printf("Enter the number of terms for Fibonacci series: ");
    scanf("%d", &terms);

    generateFibonacci(terms);

    return 0;
}
```

output:

```
Enter the number of terms for Fibonacci series: 7
Fibonacci Series up to 7 terms: 0, 1, 1, 2, 3, 5, 8,
```

Tower of Hanoi

```c
#include <stdio.h>

void towerOfHanoi(int n, char source, char auxiliary, char destination) {
    if (n == 1) {
        printf("Move disk 1 from %c to %c\n", source, destination);
        return;
    }

    towerOfHanoi(n - 1, source, destination, auxiliary);
    printf("Move disk %d from %c to %c\n", n, source, destination);
    towerOfHanoi(n - 1, auxiliary, source, destination);
}

int main() {
    int numDisks;

    printf("Enter the number of disks for Tower of Hanoi: ");
    scanf("%d", &numDisks);

    towerOfHanoi(numDisks, 'A', 'B', 'C');

    return 0;
}
```

output:

```
Enter the number of disks for Tower of Hanoi: 5
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
Move disk 4 from A to B
Move disk 1 from C to B
Move disk 2 from C to A
Move disk 1 from B to A
Move disk 3 from C to B
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 5 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
Move disk 3 from B to A
Move disk 1 from C to B
Move disk 2 from C to A
Move disk 1 from B to A
Move disk 4 from B to C
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```