**Assignment 2**

**Problem statement:**

## Recursive Function and Efficiency Analysis - Write a recursive function pseudocode and calculate the nth Fibonacci number and use Big O notation to analyze its efficiency. Compare this with an iterative approach and discuss the pros and cons in terms of space and time complexity.

**Source Code:**

## Recursive Approach:

FUNCTION fibonacci(n)

IF n <= 1

RETURN n

ELSE

RETURN fibonacci(n - 1) + fibonacci(n - 2)

In this recursive approach, each call to fibonacci(n) results in two recursive calls, fibonacci(n-1) and fibonacci(n-2), except for the base cases where the function directly returns. Therefore, the time complexity can be expressed as: O(2^n).

**Pros:**

* Simple and intuitive.
* Easy to understand and implement.

**Cons:**

* Exponential time complexity ((O(2^n))).
* High memory usage due to the call stack.

## Iterative Approach:

FUNCTION fibonacci\_iterative(n)

DECLARE F[0..n] // Array to store Fibonacci numbers

F[0] = 0

F[1] = 1

FOR i FROM 2 TO n

F[i] = F[i - 1] + F[i - 2]

RETURN F[n]

The iterative solution computes Fibonacci numbers directly without recursion. It iterates through the array once, performing constant-time operations. Hence, the time complexity is (O(n)).

**Pros:**

* Linear time complexity ((O(n))).
* Efficient use of memory.
* No risk of stack overflow.

**Cons:**

* Less straightforward for beginners.