**Assignment 3:**

Function Design and Modularization - Create a document that describes the design of two modular functions: one that returns the factorial of a number, and another that calculates the nth Fibonacci number. Include pseudocode and a brief explanation of how modularity in programming helps with code reuse and organization.

**Solution:**

**Function 1: Factorial Calculation**

function factorial(n):

if n < 0:

return "Invalid input: Factorial of negative numbers is undefined"

else if n == 0 or n == 1:

return 1;

else:

result = 1;

for i from 2 to n:

fact = fact \* i

return fact

# The time complexity of the factorial function is O(n)

# The space complexity of the factorial function is O(1)

**Code Execution:**



**Function 2: Fibonacci Number Calculation**

function fibonacci(n):

if n <= 0:

return "Invalid input: Fibonacci sequence starts from index 1"

else if n == 1 or n == 2:

return 1

else:

prev = 1

curr = 1

for i from 3 to n:

next = prev + curr

prev = curr

curr = next

return curr

# The time complexity of the fibonacci function is O(n)

# The space complexity of the fibonacci function is O(1)

**Modularity in Programming:**

Modularity in programming refers to the practice of breaking down a program into smaller, manageable, and reusable modules or functions. Here's how modularity helps with code reuse and organization:

**Code Reusability**: Modular functions can be reused in different parts of the program or even in other programs. For example, the factorial and Fibonacci functions designed above can be used wherever factorial or Fibonacci number calculation is required without rewriting the code.