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.

function factorial(n):

if n == 0 or n == 1:

return 1

else:

result := 1

for i from 2 to n:

result := result \* i

return result

If n is 0 or 1, return 1

Otherwise, initialize result to 1.

Use a loop to multiply result by each integer from 2 to n.

Return the computed factorial result.

It iterates from 1 to n, multiplying each value of i with the result.

This function is modular because it encapsulates a specific functionality

(calculating factorial) and can be reused in different parts of a program

or in other programs without rewriting the logic. By modularizing this

functionality, the code becomes more organized and easier to maintain.

function fibonacci(n):

if n == 0:

return 0

else if n == 1:

return 1

else:

prev1 := 0

prev2 := 1

current := 0

for i from 2 to n:

current := prev1 + prev2

prev1 := prev2

prev2 := current

return current

If n is 0, return 0 (base case for Fibonacci).

If n is 1, return 1 (base case for Fibonacci).

Otherwise, initialize prev1 to 0 (F(0)) and prev2 to 1 (F(1)).

Use a loop to calculate Fibonacci numbers iteratively up to n.

Update prev1, prev2, and current according to the Fibonacci sequence.

Return the computed nth Fibonacci number current.

fibonacci is modular as it encapsulates a specific functionality and can be reused

across different parts of a program or in other programs.