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.

1.function to calculate factorial:

Pseudocode:

int CalculateFactorial(int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return n \* CalculateFactorial(n - 1);

}

}

Explanation:

* The function CalculateFactorial takes an integer n as input and returns the factorial of n.
* It uses recursion to calculate the factorial by multiplying n with the factorial of n-1 until it reaches the base case where n is 0 or 1.
* This function is modular as it encapsulates the logic for calculating factorial, making it reusable and easy to understand.

2. Function to Calculate Fibonacci Number:

Pseudocode:

function CalculateFibonacci(n):

if n <= 1:

return n

else:

fib1 = 0

fib2 = 1

for i from 2 to n:

fib = fib1 + fib2

fib1 = fib2

fib2 = fib

return fib

Explanation:

* The function CalculateFibonacci takes an integer n as input and returns the nth Fibonacci number.
* It checks for the base cases where n is 0 or 1 and returns n in those cases.
* For n greater than 1, it calculates the Fibonacci number iteratively using a loop, storing the previous two Fibonacci numbers.
* This function is modular as it encapsulates the logic for calculating Fibonacci numbers, promoting code reuse and maintainability

Modularity in Programming:

* Modularity is a fundamental principle in programming that involves breaking down a program into smaller, self-contained modules or functions, each responsible for performing a specific task. This approach offers several advantages:
* Code Reusability: Modular functions can be reused in different parts of a program or even in different programs altogether, saving time and effort in coding.
* Organization: Modular design organizes code into smaller, more manageable units, making it easier to understand, maintain, and debug.
* Scalability: Modular code is easier to scale and extend as new features or functionality can be added by creating new modules or modifying existing ones without affecting the entire program.
* Abstraction: Modular functions encapsulate specific functionality, providing a clear interface for interacting with other parts of the program while hiding implementation details, which promotes better abstraction and reduces complexity.