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: Compute Factorial**

**Description:** This function takes a non-negative integer as input and returns its factorial.

**Pseudocode:**

Function factorial (n) if n is 0 return 1 else return n \* factorial (n - 1)

**Explanation:** The factorial of a non-negative integer n (denoted as n!) is the product of all positive integers less than or equal to n. In the pseudocode, the factorial function is implemented recursively. If n is 0, the function returns 1, which is the base case of the factorial function. Otherwise, it recursively calls itself with n - 1 until it reaches the base case, and then multiplies n with the result of the recursive call.

**Function 2: Calculate Fibonacci number**

**Description:** This function calculates the nth Fibonacci number.

**Pseudocode:**

Function Fibonacci (n) if n is 0 return 0 else if n is 1 return 1 else return Fibonacci (n - 1) + Fibonacci (n - 2)

**Explanation:** The Fibonacci sequence is a series of numbers in which each number is the sum of the two preceding ones, usually starting with 0 and 1. In the pseudocode, the Fibonacci function is implemented recursively. If n is 0, the function returns 0, which is the first Fibonacci number. If n is 1, the function returns 1, which is the second Fibonacci number. For n greater than 1, it recursively calls itself with n - 1 and n - 2, and returns the sum of the results.

**Modularity in Programming:** Modularity in programming refers to the practice of breaking down a program into smaller, manageable, and reusable modules or functions. Modular programming promotes code reuse, enhances maintainability, and improves code organization.

**Benefits of Modularity:**

1. **Code Reusability:** Modular functions can be reused in multiple parts of a program or in different programs altogether, reducing redundancy and promoting efficiency.
2. **Ease of Maintenance:** With modular design, each function is responsible for a specific task, making it easier to debug, update, or replace individual modules without affecting the entire program.
3. **Improved Readability:** Modular code is easier to understand and navigate because it is organized into smaller, self-contained units with well-defined purposes.
4. **Scalability:** Modularity facilitates scalability by allowing developers to add or modify functionality without disrupting the existing codebase.
5. **Collaboration:** Modular programming encourages collaboration among team members as they can work on individual modules independently and integrate them seamlessly into the larger system.

In summary, modularity enhances the quality and flexibility of software development by promoting code reuse, simplifying maintenance, and facilitating collaboration among developers.