Assignment 1: Pseudocode Development - Task: Write a detailed pseudocode for a simple program that takes a number as input, calculates the square if it's even or the cube if it's odd, and then outputs the result. Incorporate conditional and looping constructs.

Solution:

1. Start
2. Declare a variable as a 'inputNumber'
3. Print "Enter a number:"
4. Read the input from the user and store it in 'inputNumber'
5. Declare a variable as a 'result'
6. Check if 'inputNumber' is even or odd:
7. If 'inputNumber' modulo 2 equals 0:

* Set 'result' to the square of 'inputNumber'
* Print "The square of the number is: " + 'result'

1. Else:

* Set 'result' to the cube of 'inputNumber'
* Print "The cube of the number is: " + 'result'

1. Stop

Assignment 2: Flowchart Creation - Design a flowchart that outlines the logic for a user login process. It should include conditional paths for successful and unsuccessful login attempts, and a loop that allows a user three attempts before locking the account.

Solotion: Flowchart for user login process

* Start the process
* Set count attempts to 0
* Try again until attempts < 3
* Prompt user to input username and password
* Read and store the input as 'username' and 'password'
* If username and password are correct
* Display "Login Successful"
* Exit loop
* Else
* Display "Invalid username or password"
* Increment attempts by 1
* Display "Account Locked"
* End

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:

**Pseudocode**

* Start
* Initialize factorial = 1
* If n is 0 or 1
* Set factorial = 1
* Else
* Loop from i = 2 to n
* Multiply factorial by i
* Return factorial
* End

**Modular function for Fibonacci number calculation:**

**Pseudocode:**

* Start
* If n is 0
* Set fibonacciNumber = 0
* Else if n is 1
* Set fibonacciNumber = 1
* Else
* Initialize fibonacciNumber1 = 0
* Initialize fibonacciNumber2 = 1
* Loop from i = 2 to n
* Set fibonacciNumber = fibonacciNumber1 + fibonacciNumber2
* Set fibonacciNumber1 = fibonacciNumber2
* Set fibonacciNumber2 = fibonacciNumber
* Return fibonacciNumber
* End

**Modular Programming:** Modular programming is a general programming concept where developers separate program functions into independent pieces. These pieces then act like building blocks, with each block containing all the necessary parts to execute one aspect of functionality.

Modularity in programming involves breaking down a program into smaller, self-contained units (functions or modules) that perform specific tasks. This approach offers several benefits:

* 1. **Code Reusability:** Modular functions can be reused in multiple parts of a program or in different programs altogether, saving development time and effort.
  2. **Ease of Maintenance:** Since each module focuses on a specific task, it's easier to debug, update, or replace a particular module without affecting the rest of the program.
  3. **Enhanced Readability:** Modular design makes code more readable and understandable by organizing it into logical units, making it easier for developers to comprehend and maintain.
  4. **Scalability:** Modularity allows for easy scalability of a program by adding or removing modules as needed, without affecting the overall structure of the program.
  5. **Simplified Debugging:** With modularization, debugging becomes easier as issues are isolated within specific functions, allowing developers to focus on one of the code at a time.