Assignment 1: Pseudocode Development

1. 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.

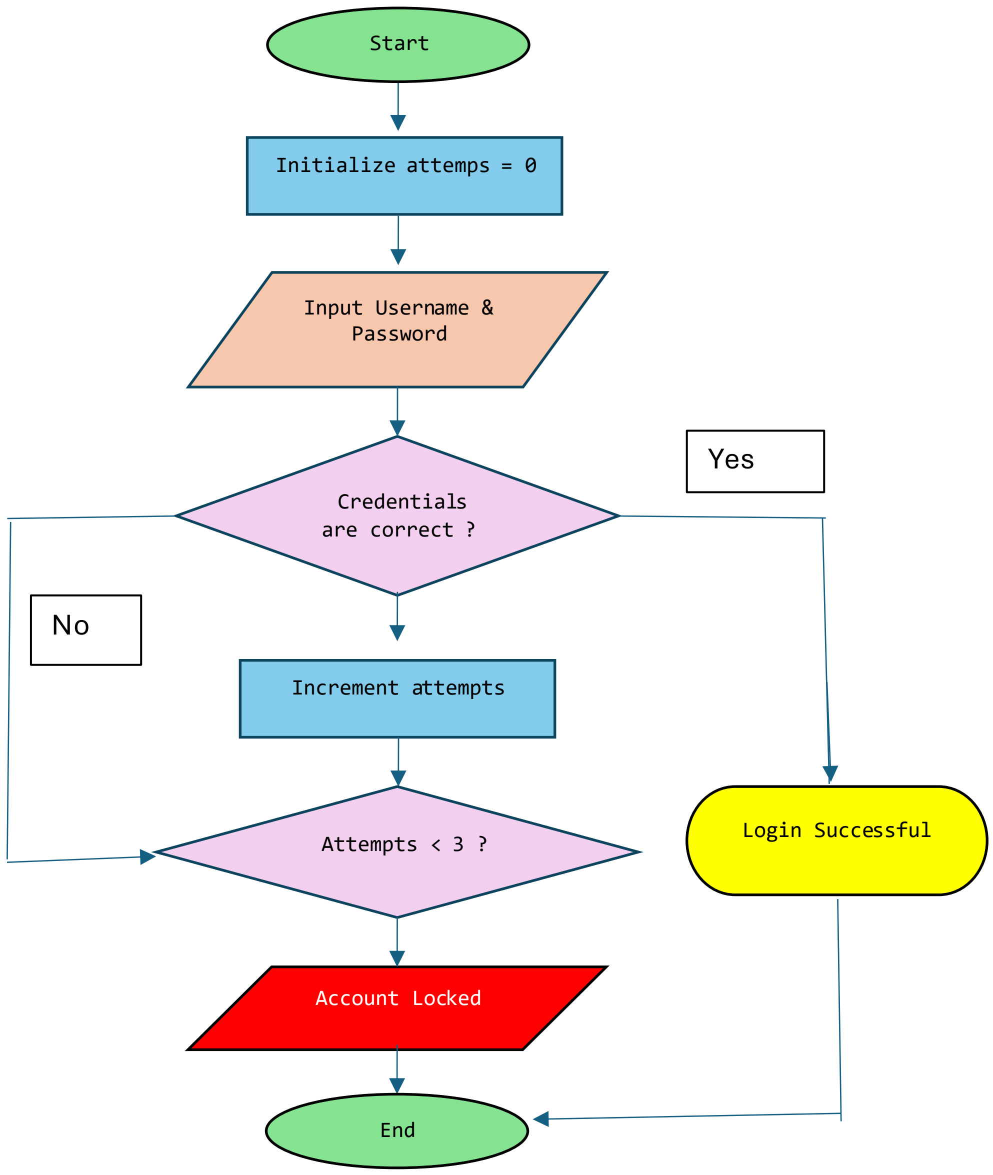
Solution:

```
BEGIN

WHILE TRUE DO
  DISPLAY "Enter a number:"
  INPUT number

  IF number MOD 2 == 0 THEN
    result ← number * number
    DISPLAY "The number is even."
    DISPLAY "Square of the number is: ", result
  ELSE
    result ← number * number * number
    DISPLAY "The number is odd."
    DISPLAY "Cube of the number is: ", result
  END IF

  DISPLAY "Do you want to continue? (yes/no):"
  INPUT choice

  IF choice == "no" OR choice == "No" THEN
    BREAK
  END IF
END WHILE

DISPLAY "Program Ended."


END
```

Assignment 2: Flowchart Creation

2. 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.

Solution:



Assignment 3: Function Design and Modularization

3. 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:

Designing Modular Functions for Factorial and Fibonacci Calculations

Modularity in Programming:

Modularity in programming refers to the practice of breaking a larger program into smaller, independent sections called modules or methods. Each module is designed to perform a specific task and can function independently. This makes the overall program easier to understand, manage, and develop.

How Modularity Helps with Code Reuse and Organization:

Modularity allows developers to isolate specific tasks into separate functions, which can be reused across programs. It also keeps the code organized, making it easier to maintain, update, and scale efficiently.

Function 1: Factorial Calculation

Purpose: Returns the factorial of a given non-negative integer.

Pseudocode:

```
FUNCTION Factorial(n)
   IF n < 0 THEN
      RETURN "Invalid input"
   ELSE IF n == 0 OR n == 1 THEN
      RETURN 1
   ELSE
      RETURN n * Factorial (n - 1)
END FUNCTION
```

Function 2: Fibonacci Number Calculation

Purpose: Returns the nth number in the Fibonacci sequence.

Pseudocode:

```
FUNCTION Fibonacci(n)
   IF n <= 0 THEN
      RETURN "Invalid input"
   ELSE IF n == 1 THEN
      RETURN 0
   ELSE IF n == 2 THEN
      RETURN 1
   ELSE
      RETURN Fibonacci (n - 1) + Fibonacci (n - 2)
END FUNCTION
```

Explanation: This recursive function returns the nth term of the Fibonacci sequence. The base cases are for the first and second terms, and for all others, it returns the sum of the two preceding terms.