Assignments:

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.

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.

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.

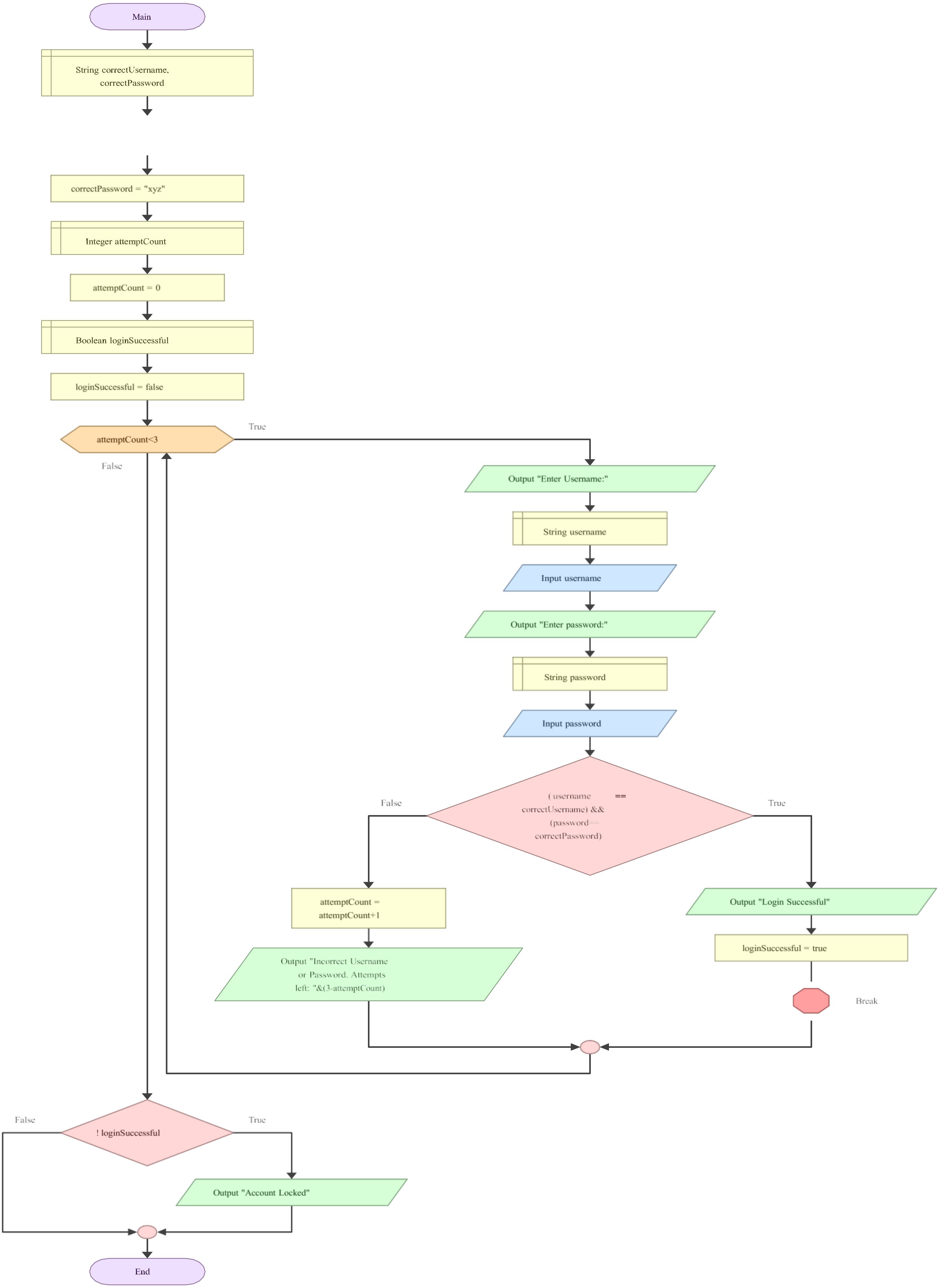
Assignment 1: Pseudocode Development - Task: Write a detailed pseudocode for a simple program that takes a number as input, calculates the square if it's even or the cube if it's odd, and then outputs the result. Incorporate conditional and looping constructs.

Solution:

```
START

REPEAT
   DISPLAY "Enter a number (or type 'exit' to quit):"
   READ input

   IF input = "exit" THEN
      EXIT LOOP
   END IF

   CONVERT input TO integer → number

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

UNTIL FALSE

DISPLAY "Program terminated."

END
```

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.

Solution:


Main
String correctUsername,
correctPassword
correctPassword = "xyz"
Integer attemptCount
attemptCount = 0
Boolean loginSuccessful
loginSuccessful = false
attemptCount<3
True
False
Output "Enter Username:"
String username
Input username
Output "Enter password:"
String password
Input password
(username == correctUsername) && (password== correctPassword)
False
True
attemptCount = attemptCount+1
Output "Incorrect Username or Password. Attempts left: "&(3-attemptCount)
Output "Login Successful"
loginSuccessful = true
Break
! loginSuccessful
False
True
Output "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:

Function Design and Modularization Document

Objective

Design two modular functions:

1. Factorial(n) – Returns the factorial of a number.
2. Fibonacci(n) – Returns the nth Fibonacci number.

What is Modularity in Programming?

Modularity means breaking a program into smaller, reusable components (functions/modules). Each function performs a specific task and can be reused in different parts of the program or in other programs.

Benefits of Modularity

- Reusability: Write once, use multiple times.
- Readability: Smaller, focused functions are easier to understand.
- Maintainability: Easier to debug and update individual parts.
- Testing: Functions can be tested independently.
- Collaboration: Multiple developers can work on different modules.

1. Factorial Function

Description:

Calculates the factorial of a non-negative integer n.

Factorial of n (written as n!) is defined as:

$n! = n \times (n-1) \times (n-2) \times ... \times 1$

Pseudocode:

```
FUNCTION Factorial(n)
    IF n < 0 THEN
        RETURN "Error: Negative number"
    ELSE IF n = 0 THEN
        RETURN 1
    END IF

    result ← 1
    FOR i FROM 1 TO n DO
        result ← result × i
    END FOR

    RETURN result
END FUNCTION
```

---

2. Fibonacci Function

Description:

Calculates the nth Fibonacci number.
Fibonacci sequence:

0, 1, 1, 2, 3, 5, 8, 13, ...

Each term is the sum of the two preceding terms:

$F(0) = 0, F(1) = 1$

$F(n) = F(n-1) + F(n-2)$

Pseudocode:

```
FUNCTION Fibonacci(n)
   IF n < 0 THEN
      RETURN "Error: Invalid input"
   ELSE IF n = 0 THEN
      RETURN 0
   ELSE IF n = 1 THEN
      RETURN 1
   END IF

   a ← 0
   b ← 1
   FOR i FROM 2 TO n DO
      temp ← a + b
      a ← b
      b ← temp
   END FOR

   RETURN b
END FUNCTION
```

Conclusion :

Using modular functions like Factorial(n) and Fibonacci(n) improves code clarity and reusability. These functions can be used in larger programs such as calculators, algorithm simulations, or educational tools without rewriting the core logic.