Q1) 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.

```
START
  REPEAT
    DISPLAY "Enter a number (or 'exit' to quit): "
    INPUT num
    IF num = "exit" THEN
      BREAK
    ENDIF
    IF num MOD 2 = 0 THEN
      DISPLAY "Even number. Square: ", num * num
    ELSE
      DISPLAY "Odd number. Cube: ", num * num * num
    ENDIF
  UNTIL FALSE
DISPLAY "Program ended."
END
```

Q3) 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.

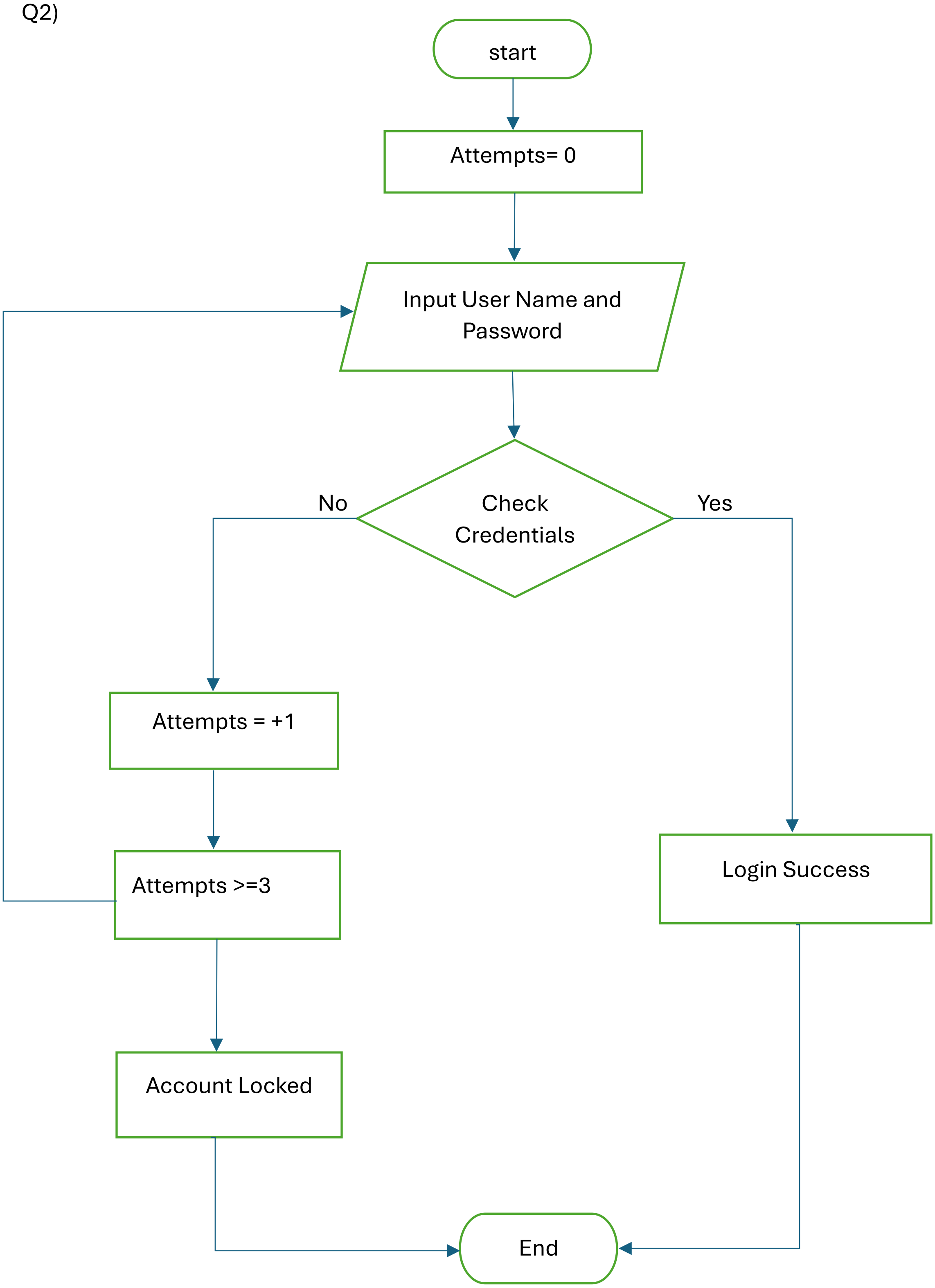
```
FUNCTION factorial(n)
  IF n == 0 OR n == 1 THEN
    RETURN 1
  ELSE
    RETURN n * factorial(n - 1)
  ENDIF
END FUNCTION
```

```
FUNCTION fibonacci(n)
  IF n == 0 THEN
    RETURN 0
  ELSE IF n == 1 THEN
    RETURN 1
  ELSE
    RETURN fibonacci(n - 1) + fibonacci(n - 2)
  ENDIF
END FUNCTION
```

Benefits of Modularity in Programming

- Code Reuse: Functions can be reused in other programs, saving development time.
- Ease of Debugging: Testing and debugging smaller modules individually reduces errors.
- Improved Readability: Breaking down large problems into functions makes the code easier to follow.
- Collaboration: Teams can work on different functions independently.

Q2)


start
Attempts= 0
Input User Name and Password
Check Credentials
No
Yes
Attempts = +1
Attempts >=3
Login Success
Account Locked
End