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.

//Algorithm:

Step1: start

Step2: get the number from user

Step3: Using If condition Check the given number is odd or even using % operator

Step4:If number is even then square the number by multiplying the number with itself

Step5: Else number is odd then cube the number by multiplying the number with itself twice.

Step6: End the program

// Pseudocode

step 1 : Start

step 2 : Getting Entered Input number

step 3: If entered Number is even, calculate the square

square = number \* number

Display "The square of", number, "is", square

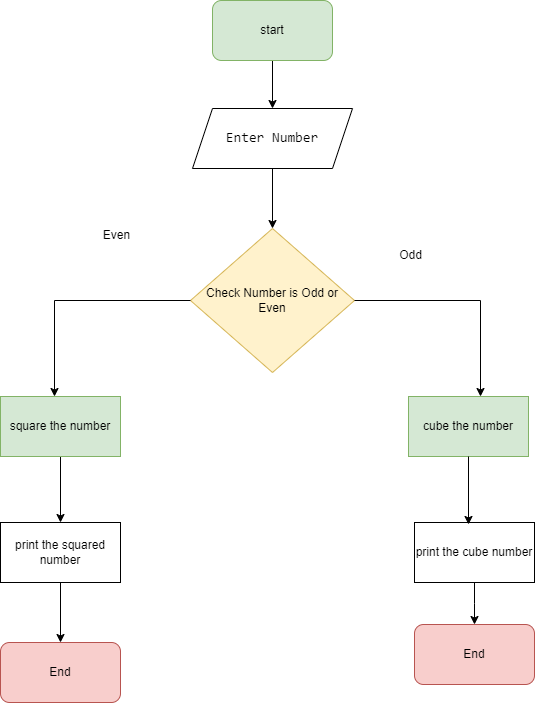
Else

cube = number \* number \* number

Display "The cube of", number, "is", cube

step 4: End of the program

Flow chart is added below



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.

//Algorithm:

Step1: start

Step2: get the Username and password from user

Step3: Using If condition Check the Username and password match the data in db

Step4: If Username and password match the data in db allow user give access to site

Step5: Else display error and repeat step2 and step 3 and if not matches display timed out and lock the account

Step6: End the program

// Pseudocode

step 1 : Start

step 2 : Getting Entered Username and password

step 3: If Username and password Matches

allow user to the site

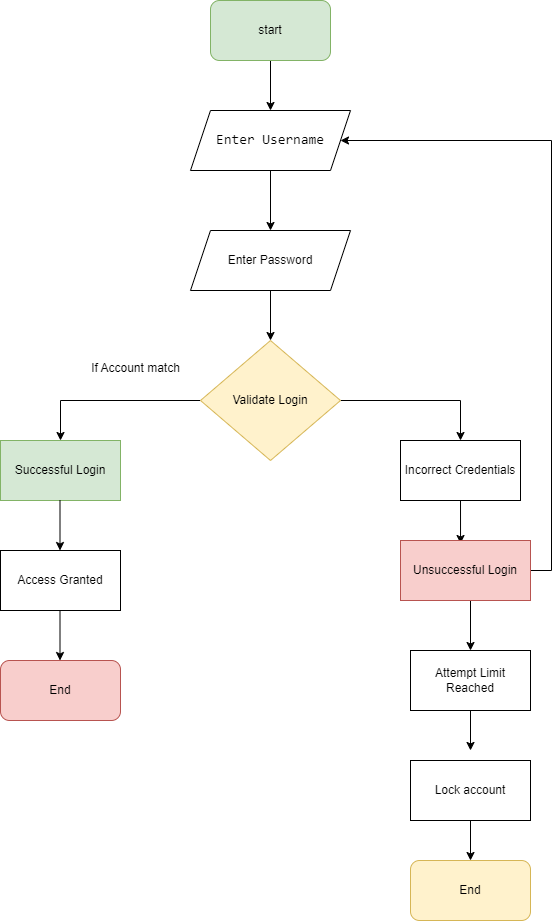
Display "Login successfully”

Else

Display "Data not fount " and repeat step 2 and step 3

step 4: End of the program

Flow chart is added below



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

//Algorithm:

Step1: start

Step2: get the number from user

Step3: Using If condition when number is 0 then return 1

Step4: Else return number\*factorial(number-1)

Step5: End the program

//Fibonacci series

Step1: start

Step2: get the number from user

Step3: Using If condition when number is 0 then return 0

Step4: Using If condition when number is 1 then return 1

Step5: Else return Fibonacci(n-1)+Fibonacci(n-2)

Step5: End the program

// pseudocode for factorial of number modular functions

1. Pseudocode:

Function factorial(number)

if number is 0

return 1

else

return number \* factorial(number - 1)

end if

End Function

Explanation: This function recursively calculates the factorial of a number. If the input number is 0, it returns 1 (since the factorial of 0 is defined as 1). Otherwise, it recursively multiplies the number with the factorial of (number - 1) until it reaches the base case (number = 0).

// modular function for Fibonacci number.

2.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)

end if

End Function

Explanation: This function recursively calculates the nth Fibonacci number. If the input n is 0, it returns 0. If n is 1, it returns 1. Otherwise, it recursively calls itself to calculate the (n-1)th and (n-2)th Fibonacci numbers, and returns their sum.

Modularity in Programming:

Modularity in programming refers to the practice of breaking down a program into smaller, independent modules or functions. Each module performs a specific task, and can be developed, tested, and maintained separately.

