**Q1. Write a detailed pseudocode for a simple program that takes a number as input calculates the square if it is even or the cube if it is odd and then outputs the result. Incorporate conditional and lopping constructs**.

**Procedure:**

Step1 - Start

Step2 - Enter a number and store it in a variable 'num'

Step2 - Check if the number is even or odd:

Step4 - If 'num' modulo 2 is equal to 0 (i.e., num % 2 == 0), it's even:

Step5 - Calculate the square of 'num' and store it in a variable 'result'

Step6 - Else, it's odd

Step7 - Calculate the cube of 'num' and store it in a variable 'result'

Step8 - Output the result

Step9 - Ask the user if they want to calculate for another number:

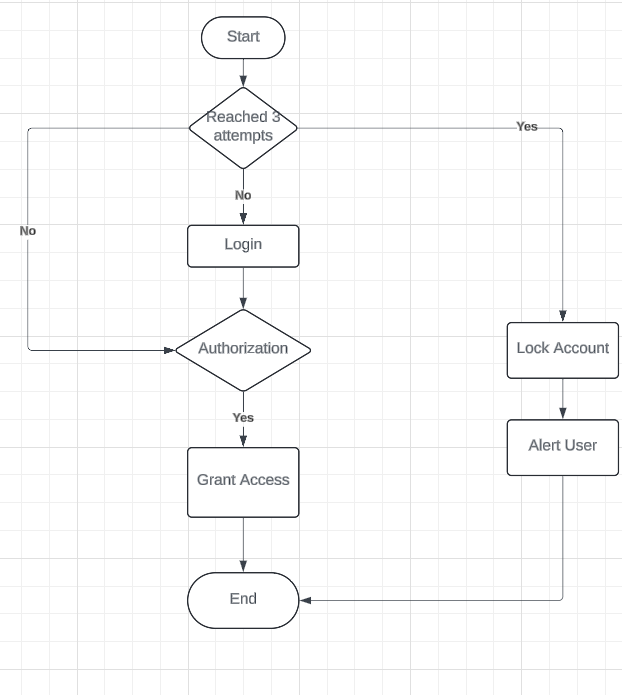
--> If yes, go back to step 2

--> If no, end the program

Step10 - End

**Q2. Design a flowchart that outlines the logic for a user login process. It should include conditional paths for successful and unsuccessful login attempts and loop that allows a user three attempts before logging the account.**

**Flowchart**:



**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.**

**Pseudocode:**

Step1: Start

Step2: Initialize a variable called result to 1

Step3: in n=0 or n=1

Step4: return 1

Step5: Otherwise, for I from 2 to n

Multiply the result by i

Step6: return value

Step7: End

Factorial of the number:

Function: factorial(n)

Input: integer n

Output: factorial of n

**Assignemnt4:**

**Pseudocode and flowchart for sorting algorithm: Write Pseudocode and create a flowchart for a bubble sort algorithm. Provide a brief explanation of how the algorithm works and a simple array of integers to demonstrate a dry run of your algorithm.**

**Pseudocode:**

Step1: Start

Step2: Read the array of given items from the user

Step3: Take the first element (index=0), compare the current element with the next element

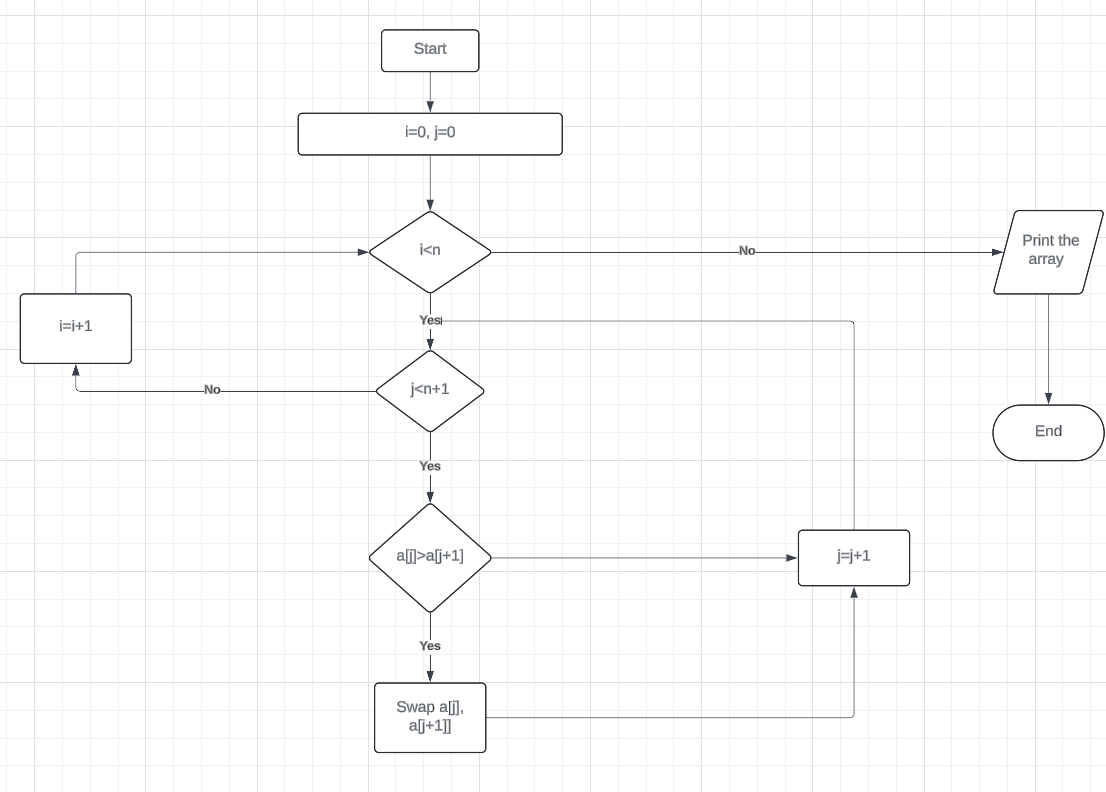
Step5: Else,

If the current element is less than the next element, then move to the next element

Step6: Repeat Step3 to 5 until all elements are sorted

Step7: Stop

**Flowchart:**

****

**Assignment4: Recursive function and efficiency analysis: Write a recursive function pseudocode and calculate the nth Fibonacci number and use Big O notation to analyze its efficiency. Compare this with an Iterative approach and discuss the pros and cons in terms of space and time complexity.**

int recursiveFun1(int n)

{

If (n<=0)

return 1;

else

return 1 + recursiveFun(n-1);

}

int recursiveFun2(int n)

{

If (n<=0)

return 1;

else

return1 + recursiveFun2(n-2);

}

int recursiveFun3(int n)

{

if(n<=0)

return1;

else

return1 + recursiveFun3(n/5);

}

void recursiveFun4(int n, int m, int o)

{

if (n<=0)

{

Print (“%d, %d\n”, m, o);

}

else

{

recursiveFun4(n-1, m+1, o);

recursiveFun4(n-1, m, o+1);

}

}

int recursiveFun5(int n)

{

for (i=0, i<n, i+=2){

}

if (n<=0)

return1;

else

return1 + recursiveFun5(n-5);

}