**Assignment 1:** 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.

Solution:

**Pseudocode:**

1. Start
2. function bubbleSort(array):
3. n = length of array
4. for i from 0 to n - 1:
5. for j from 0 to n - i - 1:
6. if array[j] > array[j + 1]:
7. swap(array[j], array[j + 1])
8. return array
9. End

**Flowchart:**

Start

|

V

Initialize n as length of array

|

V

Set i = 0 Loop while i < n - 1

|

V

Set j = 0 Loop while j < n - i – 1

|

V

If array[j] > array[j + 1], swap

|

V

Increment j

|

V

Increment i

|

V

Return sorted

|

V

End

**Dry Run:**

For example->

Array: [5, 2, 9, 1, 6]

1st Case- Array: [2, 5, 1, 6, 9]

2nd Case- Array: [2, 1, 5, 6, 9]

3rd Case- Array: [1, 2, 5, 6, 9]

Sorted Array is Array: [5, 2, 9, 1, 6]

**Assignment 2:** 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.

Solution:

**Psuedocode:**

Function FibonacciRecursive(n):

if n <= 1 then

return n

else

return FibonacciRecursive(n - 1) + FibonacciRecursive(n - 2)

end if

End Function

**->**This recursive function calculates the nth Fibonacci number by recursively calling itself with

n-1 and n-2 until reaching the base cases (n=0 or n=1).

**Recursive Approach:**

* Pros: The recursive approach is often more straightforward and easier to understand, especially for those familiar with recursive algorithms.
* Cons: The recursive approach suffers from exponential time complexity, making it inefficient for large values of n. and Recursion can lead to stack overflow errors for large values of n due to the limited size of the call stack.

**Iterative Approach:**

* Pros: Linear time complexity (O(n)), making it more efficient than the recursive approach for large n. Lower memory usage as it does not rely on function call stack.
* Cons: The iterative solution may involve more complex code compared to the recursive solution, especially for those less familiar with iterative algorithms.