Data Structures and Operations [Day 4] :

Task 1: Array Sorting and Searching

a) Implement a function called BruteForceSort that sorts an array using the brute force approach. Use this function to sort an array created with InitializeArray.

```
package com.wipro.assignment03;
import java.util.Random;
public class ArraySorter
{
    // Function to initialize an array with random integers
    public static int[] InitializeArray(int size, int lowerBound, int upperBound) {
        Random rand = new Random();
        int[] array = new int[size];
        for (int i = 0; i < size; i++) {
            array[i] = rand.nextInt((upperBound - lowerBound) + 1) + lowerBound;
        }
        return array;
    }

    // Brute force sort function
    public static void BruteForceSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                if (arr[i] > arr[j]) {
                    // Swap arr[i] and arr[j]
                    int temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
        }
    }

    // Helper function to print the array
    public static void printArray(int[] arr)
    {
        for (int value : arr) {
            System.out.print(value + " ");
        }
        System.out.println();
    }

    // Main method to demonstrate the functions
    public static void main(String[] args)
    {
        // Initialize an array
        int[] array = InitializeArray(10, 0, 50);
        System.out.println("Original array:");
        printArray(array);

        // Sort the array using brute force sort
```

```
        BruteForceSort(array);
        System.out.println("Sorted array:");
        printArray(array);
    }
}
```

Output:

```
Original array:
28 24 40 23 50 24 8 32 45 50
Sorted array:
8 23 24 24 28 32 40 45 50 50
```

b) Write a function named Perform Linear Search that searches for a specific element in an array and returns the index of the element if found or -1 if not found.

```
package com.wipro.assignment03;
public class ArraySearcher
{
    // Linear search function
    public static int PerformLinearSearch(int[] arr, int target) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == target) {
                return i;
            }
        }
        return -1;
    }

    // Helper function to print the array
    public static void printArray(int[] arr)
    {
        for (int value : arr) {
            System.out.print(value + " ");
        }
        System.out.println();
    }

    // Main method to demonstrate the function
    public static void main(String[] args)
    {
        // Initialize an array
        int[] array = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
        System.out.println("Array:");
        printArray(array);

        // Perform linear search for a specific element
        int target = 80;
        int index = PerformLinearSearch(array, target);
        System.out.println("Element " + target + " found at index: " + index);

        // Test with an element not in the array
```

```
        target = 55;
        index = PerformLinearSearch(array, target);
        System.out.println("Element " + target + " found at index: " + index);
    }
}
```

Output:

```
Array:
10 20 30 40 50 60 70 80 90 100
Element 80 found at index: 7
Element 55 found at index: -1
```

Task 2: Two-Sum Problem

a) Given an array of integers, write a program that finds if there are two numbers that add up to a specific target. You may assume that each input would have exactly one solution, and you may not use the same element twice. Optimize the solution for time complexity.

```
package com.wipro.assignment03;
import java.util.HashMap;
import java.util.Map;
public class TwoSumSolution
{

    // Function to find two numbers that add up to a specific target
    public static int[] findTwoSum(int[] nums, int target)
    {
        Map<Integer, Integer> numToIndexMap = new HashMap<>();
        for (int i = 0; i < nums.length; i++) {
            int complement = target - nums[i];
            if (numToIndexMap.containsKey(complement)) {
                return new int[] { numToIndexMap.get(complement), i };
            }
            numToIndexMap.put(nums[i], i);
        }
        throw new IllegalArgumentException("No two sum solution");
    }
    // Main method to demonstrate the function
    public static void main(String[] args)
 {
        int[] array = {2, 7, 11, 15};
        int target = 9;
        int[] result = findTwoSum(array, target);
        System.out.println("Indices of the two numbers are: " + result[0] + " and " +
result[1]);
    }
}
```

Output:

```
Indices of the two numbers are: 0 and 1.
```

Task 3: Understanding Functions through Arrays

a) Write a recursive function named SumArray that calculates and returns the sum of elements in an array, demonstarte with example.

```
package com.wipro.assignment03;
public class SumArrayRecursive
{

    // Recursive function to calculate the sum of elements in an array
    public static int SumArray(int[] arr, int n)
    {
        // Base case: if the array is empty or we've processed all elements
        if (n <= 0) {
            return 0;
        }
        // Recursive case: sum the last element and the sum of the rest of the array
        return arr[n - 1] + SumArray(arr, n - 1);
    }

    // Main method to demonstrate the function
    public static void main(String[] args)
    {
        int[] array = {1, 2, 3, 4, 5,10};
        // Calculate the sum of the array elements using the recursive function
        int sum = SumArray(array, array.length);
        // Print the sum
        System.out.println("Sum of array elements: " + sum);
    }
}
```

Output:

```
Sum of array elements: 25
```

Task 4: Advanced Array Operations

a) Implement a method SliceArray that takes an array, a starting index, and an end index, then returns a new array containing the elements from the start to the end index.

```
 package com.wipro.assignment03;
import java.util.Arrays;
public class ArraySlicer
{
    // Method to slice the array
    public static int[] SliceArray(int[] arr, int startIndex, int endIndex) {
        // Check for valid indices
```

```
        if (startIndex < 0 || endIndex >= arr.length || startIndex > endIndex) {
            throw new IllegalArgumentException("Invalid start or end index");
        }

        // Calculate the length of the new array
        int[] slicedArray = new int[endIndex - startIndex + 1];

        // Copy elements from the original array to the new array
        for (int i = startIndex; i <= endIndex; i++) {
            slicedArray[i - startIndex] = arr[i];
        }

        return slicedArray;
    }

    // Main method to demonstrate the function
    public static void main(String[] args)
    {
        int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

        // Define start and end indices
        int startIndex = 2;
        int endIndex = 5;

        // Get the sliced array
        int[] slicedArray = SliceArray(array, startIndex, endIndex);

        // Print the sliced array
        System.out.println("Sliced array: " + Arrays.toString(slicedArray));
    }
}
```

Output:

```
Sliced array: [3, 4, 5, 6]
```

b) Create a recursive function to find the nth element of a Fibonacci sequence and store the first n elements in an array.

```
package com.wipro.assignment03;
import java.util.Arrays;
public class FibonacciRecursive
{
```

```
    // Recursive function to find the nth Fibonacci number
    public static int fibonacci(int n) {
        if (n <= 1) {
            return n;
        }
        return fibonacci(n - 1) + fibonacci(n - 2);
    }

    // Function to store the first n Fibonacci numbers in an array
    public static int[] fibonacciArray(int n) {
        int[] fibArray = new int[n];
        for (int i = 0; i < n; i++) {
            fibArray[i] = fibonacci(i);
        }
        return fibArray;
    }

    // Main method to demonstrate the functions
    public static void main(String[] args)
    {
        int n = 10; // Change this value to generate more or fewer Fibonacci numbers

        // Get the first n Fibonacci numbers
        int[] fibArray = fibonacciArray(n);

        // Print the Fibonacci array
        System.out.println("The first " + n + " elements of the Fibonacci sequence
are:");
        System.out.println(Arrays.toString(fibArray));
    }
}
```

Output:

```
The first 10 elements of the Fibonacci sequence are:
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```