CCC2113 Data Structures and Algorithms Analysis Name:

Lab 5: Search Algorithms (16/5/2024) Group:

1. Write a method `linearSearch` that takes an array of integers and a target integer. The method should return the index of the target if found in the array, or return -1 if not found. [2 marks]

```java
public class LinearSearchExample{
    public static int LinearSearch(int[] arr, int target){
        for(int i =0; i < arr.length; i++){
            if(target == arr[i]){
                return i;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50};
        int target = 30;
        int result = LinearSearch(numbers, target);

        if (result != -1) {
            System.out.println("Target found at index: " + result);
        } else {
            System.out.println("Target not found in the array.");
        }
    }
}
```

2. Write a method `binarySearch` that takes an array of integers and a target integer. The method should return the index of the target if found in the array, or return -1 if not found.

```java
public class BinarySearchExample {
    public static int binarySearch(int[] array, int target) {
        int left = 0;
        int right = array.length - 1;

        while (left <= right) {
            int middle = left + (right - left) / 2;


            if (array[middle] == target) {
                return middle;
            }

            // If the target is greater, ignore the left half
            if (array[middle] < target) {
                left = middle + 1;
            } else {
                // If the target is smaller, ignore the right half
                right = middle - 1;
            }
        }

        return -1;
    }

    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50};
        int target = 30;
        int result = binarySearch(numbers, target);

        if (result != -1) {
            System.out.println("Target found at index: " + result);
        } else {
            System.out.println("Target not found in the array.");
        }
    }
}
```

3. Write a method `findAll` that takes an array of integers and a target integer. The method should return a list of all indices where the target is found in the array. If the target is not found, return -1.

   Example:

   Input: array = [1, 2, 3, 2, 4, 2, 5] and target = 2 [3 marks]

   Output: *1*
   3
   5

```java
import java.util.ArrayList;
import java.util.List;

class FindAll{
    public static List<Integer> findAll(int[] arr, int target){
        List<Integer> found = new ArrayList<>();

        // Iterate through the array
        for (int i = 0; i < arr.length; i++) {
            // If the target is found, add the index to the list
            if (arr[i] == target) {
                found.add(i);
            }
        }

        // If the target is not found, return a list containing -1
        if (found.isEmpty()) {
            found.add(-1);
        }

        return found;
    }

    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 2, 4, 2, 5};
        int target = 2;
        List<Integer> result = findAll(numbers, target);

        // Print the indices where the target is found
        for (int index : result) {
            System.out.println(index);
        }
    }
}
```

4. Write a method `insertPosition` that takes a sorted array of integers and a target integer. The method should return the index at which the target should be inserted to maintain the sorted order.

   Example:

   Input: array = [1, 2, 3, 5, 6, 7] and target = 4 [3 marks]

   Output: *3*

   Input: array = [2, 3, 5, 6, 7] and target = 1

   Output: *0*

```java
public class InsertPositionExample {
    public static int insertPosition(int[] array, int target) {
        int left = 0;
        int right = array.length - 1;

        while (left <= right) {
            int middle = left + (right - left) / 2;

            if (array[middle] == target) {
```

```
                return middle;
            } else if (array[middle] < target) {
                left = middle + 1;
            } else {
                right = middle - 1;
            }
        }

        // If the target is not found, left will be the insertion position
        return left;
    }

    public static void main(String[] args) {
        int[] numbers1 = {1, 2, 3, 5, 6, 7};
        int target1 = 4;
        int result1 = insertPosition(numbers1, target1);
        System.out.println("Insert position for target " + target1 + " is: " + result1);
    }
}
```