JAVA Logic Placement Preparation Test 4

SET 1

Q.1) Print unique sorted array ⟍Accept data in sorted order having duplicate value. You need to print unique array using single loop . Unique sorted array using 1 loop Input⟍ 1 1 2 2 2 5 output⟍1 2 5

Ans :-

```
public class Q1 {
  public static void printUnique(int arr[]){


    int n = arr.length;


    System.out.print(arr[0]);


    for(int i = 1; i < n; i++){
      if(arr[i] != arr[i-1]){
        System.out.print(" " + arr[i]);
      }
    }
    System.out.println();
  }
  public static void main(String[] args) {
    int arr[] = {1, 1, 2, 2, 2, 5};
    printUnique(arr);
  }
}
```

Q.2) To find the maximum sum of all subarrays of size K: Given an array of integers of size ‘n’, Our aim is to calculate the maximum sum of ‘k’ consecutive elements in the array. Input : arr[] = {100, 200, 300, 400}, k = 2 Output : 700

Ans :-

```
public class Q2 {

   public static int sum_of_all_subarray(int arr[], int k){

      int n = arr.length;
      int max_sum = 0;
      for(int i = 0; i < k; i++){
         max_sum += arr[i];
      }
      int window_sum = max_sum;
      for(int i = k; i < n; i++){
         window_sum += arr[i] - arr[i-k];
         max_sum = Math.max(max_sum, window_sum);
      }
      return max_sum;

   }
```

SET 2

Q.1) Print unique sorted array ⎝Accept data in sorted order having duplicate value. You need to print unique array using single loop . Unique sorted array using 1 loop Input⎝ 1 1 2 2 2 5 output⎝1 2 5

Ans :-

```
public class Q1 {
   public static void printUnique(int arr[]){

      int n = arr.length;

      System.out.print(arr[0]);
```

```
        for(int i = 1; i < n; i++){
            if(arr[i] != arr[i-1]){
                System.out.print(" " + arr[i]);
            }
        }
        System.out.println();
    }
    public static void main(String[] args) {
        int arr[] = {1, 1, 2, 2, 2, 5};
        printUnique(arr);
    }
}
```

Q.2) Given a 1-based indexing array arr[] of non-negative integers and an integer sum. You mainly need to return the left and right indexes(1-based indexing) of that subarray. In case of multiple subarrays, return the subarray indexes which come first on moving from left to right. If no such subarray exists return an array consisting of element -1. Examples: Input: arr[] = [15, 2, 4, 8, 9, 5, 10, 23], target = 23 Output: [2, 5] Explanation: Sum of subarray arr[2...5] is 2 + 4 + 8 + 9 = 23. Input: arr[] = [1, 10, 4, 0, 3, 5], target = 7 Output: [3, 5] Explanation: Sum of subarray arr[3...5] is 4 + 0 + 3 = 7. Input: arr[] = [1, 4], target = 0 Output: [-1] Explanation: There is no subarray with 0 sum.

Ans :-

```
public class Q2Set2 {


    public static int[] subarray(int arr[], int target){
        int n = arr.length;
        int right = 0;
        int left = 0;
        int curr_sum = 0;
        while(right < n){
            curr_sum += arr[right];



            while (curr_sum > target && left <= right) {
                curr_sum -= arr[left];
                left ++;
            }
            if(curr_sum == target){
                return new int[] {left+1, right+1};
            }
            right++;
        }
        return new int[] {-1};
    }
    public static void main(String[] args) {
        int arr[] = {15, 2, 4, 8, 9, 5, 10, 23};
        int target = 23;
        int[] res = subarray(arr, target);
        System.out.println(res);
    }
}
```