ASSIGNMENT-2 (ARRAYS)

SOLUTIONS:-

Q1.

```java
import java.util.*;

public class arrays2_assig1 {

    static int arrayPairSum(int[] nums) {
        int n = nums.length;
        int sum = 0;
        Arrays.sort(nums);
        for (int i = 0; i < n; i += 2) {
            sum += nums[i];
        }
        return sum;
    }

    public static void main(String[] args) {
        int[] nums = { 1, 4, 3, 2 };
        int result = arrayPairSum(nums);
        System.out.println("Result: " + result);

    }
}
```

Q2.

```java
import java.util.*;

public class arrays2_assig2 {

    static int distributeCandies(int[] candyType) {
        Set<Integer> s = new HashSet<>();
        for (int c : candyType) {
            s.add(c);
        }
        return Math.min(candyType.length / 2, s.size());
    }

    public static void main(String[] args) {
        int[] candyType = { 1, 1, 2, 2, 3, 3 };
        int result = distributeCandies(candyType);
        System.out.println(result);
    }
```

```
}
```

Q3.

```
import java.util.*;

public class arrays3_assig3 {

    static int findLHS(int[] nums) {
        int ans = 0;
        Map<Integer, Integer> count = new HashMap<>();

        for (final int num : nums)
            count.merge(num, 1, Integer::sum);

        for (final int num : count.keySet())
            if (count.containsKey(num + 1))
                ans = Math.max(ans, count.get(num) + count.get(num + 1));

        return ans;
    }

    public static void main(String[] args) {
        int[] nums = { 1, 3, 2, 2, 5, 2, 3, 7 };
        int result = findLHS(nums);
        System.out.println("The longest harmonious subsequence is: " +
result);
    }
}
```

Q4.

```
public class arrays2_assig4 {

    static boolean canPlaceFlower(int[] flowerbed, int n) {
        int total = 0;
        for (int i = 0; i < flowerbed.length; i++) {
            if (flowerbed[i] == 0) {
                int prev = i == 0 ? 0 : flowerbed[i - 1];
                int next = i == 0 ? 0 : flowerbed[i + 1];
                if (prev == 0 && next == 0) {
                    total++;
                }
            }
```

```
        }
        return total == n;
    }

    public static void main(String[] args) {
        int[] flowerbed = { 1, 0, 0, 0, 1 };
        int n = 1;
        if (canPlaceFlower(flowerbed, n)) {
            System.out.println("yes");
        } else {
            System.out.println("No");
        }
    }
}
```

Q5.

```
public class arrays2_assig5 {

    static int maximumProduct(int[] nums) {
        int max1 = Integer.MIN_VALUE;
        int max2 = max1;
        int max3 = max1;

        int min1 = Integer.MAX_VALUE;
        int min2 = min1;

        for (int i = 0; i < nums.length; i++) {
            int val = nums[i];
            if (val >= max1) {
                max3 = max2;
                max2 = max1;
                max1 = val;
            } else if (val >= max2) {
                max3 = max2;
                max2 = val;
            } else if (val > max3) {
                max3 = val;
            }
            if (val < min1) {
                min2 = min1;
                min1 = val;
            } else if (val < min2) {
                min2 = val;
            }

        }
```

```
        return Math.max(min1 * min2 * max1, max1 * max2 * max3);
    }

    public static void main(String[] args) {
        int[] nums = { 1, 2, 3 };
        int result = maximumProduct(nums);
        System.out.println("Maximum product of three numbers from array is: "
+ result);
    }
}
```

Q6.

```
public class arrays2_assig6 {

    static int searchElement(int[] arr, int target) {
        int s = 0;
        int e = arr.length;
        int idx = -1;
        while (s <= e) {
            int mid = s + (e - s) / 2;
            if (arr[mid] == target) {
                idx = mid;
                return mid;
            } else if (target > arr[mid]) {
                s = mid + 1;
            } else {
                e = mid - 1;
            }
        }
        return idx;
    }

    public static void main(String[] args) {
        int[] arr = { -1, 0, 3, 5, 9, 12 };
        int target = 9;
        int idx = searchElement(arr, 9);

        System.out.println("Index where element is found is: " + idx);
    }
}
```

Q7.

```
public class arrays2_assig7 {
```

```
    static boolean isMonotonic(int[] nums) {
        boolean increasing = true;
        boolean decreasing = true;

        for (int i = 1; i < nums.length; ++i) {
            increasing &= nums[i] >= nums[i - 1];
            decreasing &= nums[i] <= nums[i - 1];
        }

        return increasing || decreasing;
    }

    public static void main(String[] args) {
        int nums[]={1,2,2,3};
        System.out.println(isMonotonic(nums));
    }
}
```

Q8.

```
public class arrays2_assig8 {

    static int smallestRangeI(int[] A, int K) {
        int min = A[0], max = A[0];

        for (int x : A) {
            min = Math.min(min, x);
            max = Math.max(max, x);
        }
        return Math.max(0, max - min - 2 * K);
    }

    public static void main(String[] args) {
        int[] nums = { 0, 10 };
        int k = 2;
        int result = smallestRangeI(nums, k);
        System.out.println(result);
    }
}
```