Assignment No.2

Q.1 Answer:

```
public class Main {
    public static void main(String[] args) {
        ArrayPairSum arrayPairSum = new ArrayPairSum();
        int[] nums = {1, 4, 3, 2};
        int result = arrayPairSum.arrayPairSum(nums);
        System.out.println(result);
    }
}
```

Q.2 Answer:

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

Q.3.Answer:

```
import java.util.HashMap;
import java.util.Map;
public class LongestHarmoniousSubsequence {
   public static int findLHS(int[] nums) {

      Map<Integer, Integer> count = new HashMap < >();
    for (int num : nums) {
         count.put(num, count.getOrDefault(num, 0) + 1);
      }
      int maxLen = 0;
      for (int num : count.keySet()) {
         if (count.containsKey(num + 1)) {
            int length = count.get(num) + count.get(num + 1);
            maxLen = Math.max(maxLen, length);
         }
      }
      return maxLen;
   }

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

Q.4Answer:

```
 class PlaceFlowers {
    public static boolean PlaceFlowers(int[] flowerbed, int n) {
        int count = 0;
        int length = flowerbed.length;
        int i = 0;

        while (i < length) {
            if (flowerbed[i] == 0 && (i == 0 || flowerbed[i - 1] == 0) && (i == length - 1 ||
flowerbed[i + 1] == 0)) {
                flowerbed[i] = 1;
                count++;
            }

            if (count >= n) {
                return true;
            }
            i++;
        }
        return false;
    }

    public static void main(String[] args) {
        int[] flowerbed = {1, 0, 0, 0, 1};
        int n = 1;
        boolean Place = PlaceFlowers(flowerbed, n);
        System.out.println("place flowers: " + Place);
    }
}
```

Q.5.Asnwer:

```
import java.util.Arrays;

public class MaximumProduct {
    public static int maximumProduct(int[] nums) {
        Arrays.sort(nums);

        int n = nums.length;



        // 1st Case :
        int maxProduct = nums[n - 1] * nums[n - 2] * nums[n - 3];
        // then
        // 2nd Case :
        int altProduct = nums[0] * nums[1] * nums[n - 1];

        // Return two cases of the maximum
        return Math.max(maxProduct, altProduct);
    }

    public static void main(String[] args) {
        int[] nums = {1, 2, 3};
        int maxProduct = maximumProduct(nums);
        System.out.println("Maximum product: " + maxProduct);
    }
}
```

Q.6 Answer:

```
public class BinarySearch {
    public static int search(int[] nums, int target) {
        int left = 0;
        int right = nums.length - 1;

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

            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return -1;
    }

    public static void main(String[] args) {
        int[] nums = {-1, 0, 3, 5, 9, 12};
        int target = 9;
        int result = search(nums, target);
        System.out.println(result);
    }
}
```

Q.7 Answer:

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

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

        return increasing || decreasing;
    }

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

Q.8.Answer

```
import java.util.*;

public class MinScore {
    public static int minScore(int[] nums, int k) {
        if (nums.length == 1) {
            return 0;
        }

        Arrays.sort(nums); // Sort the array in ascending order
        int n = nums.length;
        int max = nums[n - 1];
        int min = nums[0];
        int diff = max - min;

        if (diff <= 2 * k) {
              return 0;
        } else {

            return diff - 2 * k;
        }
    }

    public static void main(String[] args) {
        int[] nums = {1};
        int k = 0;
        int minScore = minScore(nums, k);
        System.out.println("Minimum Score: " + minScore);
    }
}
```