Assignment day-1

NAME :- SUSHANT KUMAR SINGH
Email:- ssushant886@gmail.com

Question-1

Find a pair with the given sum in an array
Given an unsorted integer array, find a pair with the given sum in it.
For example
Input: nums = [8, 7, 2, 5, 3, 1]target = 10 Output: Pair found (8, 2)orPair found (7, 3)

Answer :-

```
 import java.util.Scanner;
public class HelloWorld {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the size of the array: ");
      int n = scanner.nextInt();
      int[] nums = new int[n];
      System.out.println("Enter the elements of the array:");
      for (int i = 0; i < n; i++) {
         nums[i] = scanner.nextInt();
      }
      System.out.print("Enter the target sum: ");
      int target = scanner.nextInt();
      boolean pairFound = false;
      for (int i = 0; i < n - 1; i++) {
         int firstNum = nums[i];
         for (int j = i + 1; j < n; j++) {
            int secondNum = nums[j];
            if (firstNum + secondNum == target) {
               System.out.println("Pair found (" + firstNum + ", " + secondNum + ")");
               pairFound = true;
            }
         }
  }
      if (!pairFound) {
         System.out.println("No pair found with the given sum.");
      }
  scanner.close();
   }
}
```

Question-2

Given an integer array, replace each element with the product of every other element without using the division operator.
For example,
Input: { 1, 2, 3, 4, 5 }Output: { 120, 60, 40, 30, 24 } Input: { 5, 3, 4, 2, 6, 8 }Output: { 1152, 1920, 1440, 2880, 960, 720 }

Answer:-

```
import java.util.Scanner;
public class HelloWorld {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the size of the array: ");
      int n = scanner.nextInt();
      int[] nums = new int[n];
      System.out.println("Enter the elements of the array:");
      for (int i = 0; i < n; i++) {
         nums[i] = scanner.nextInt();
      }
      int[] result = new int[n];
      int leftProduct = 1;
      for (int i = 0; i < n; i++) {
         result[i] = leftProduct;
         leftProduct *= nums[i];
      }
      int rightProduct = 1;
      for (int i = n - 1; i >= 0; i--) {
         result[i] *= rightProduct;
         rightProduct *= nums[i];
      }
      System.out.print("Output: { ");
      for (int i = 0; i < n; i++) {
         System.out.print(result[i]);
         if (i < n - 1) {
            System.out.print(", ");
         }
      }
      System.out.println(" }");

      scanner.close();
   }
}
```

Question-3

Maximum Sum Circular Subarray
Given a circular integer array, find a subarray with the largest sum in it.
For example :Input: {2, 1, -5, 4, -3, 1, -3, 4, -1} Output: Subarray with the largest sum is {4, -1, 2, 1} with sum 6.

Answer:-

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

public class MaxCircularSubarraySum {

   public static int maxSubarraySumCircular(List<Integer> A) {
      int totalSum = 0, currSum1 = 0, currSum2 = 0, maxSumSubarray = Integer.MIN_VALUE, minSumSubarray =
Integer.MAX_VALUE;

      for (int i : A) {
         totalSum += i;
         currSum1 += i;
         currSum2 += i;

         maxSumSubarray = Math.max(maxSumSubarray, currSum1);
         if (currSum1 < 0) currSum1 = 0;

         minSumSubarray = Math.min(currSum2, minSumSubarray);
         if (currSum2 > 0) currSum2 = 0;
      }

      return (totalSum == minSumSubarray) ? maxSumSubarray : Math.max(maxSumSubarray, totalSum -
minSumSubarray);
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the size of the array: ");
      int n = scanner.nextInt();

      List<Integer> A = new ArrayList<>();
      System.out.println("Enter the elements of the array:");
      for (int i = 0; i < n; i++) {
         A.add(scanner.nextInt());
      }

      int result = maxSubarraySumCircular(A);
      System.out.println("Maximum circular subarray sum is: " + result);

      scanner.close();
   }
}
```

Question-4:

Find the maximum difference between two array elements that satisfies the given constraints
Given an integer array, find the maximum difference between two elements in it such that the smaller element appears before the larger element.
For example:Input: { 2, 7, 9, 5, 1, 3, 5 } Output: The maximum difference is 7. The pair is (2, 9)

Answer:-

```
import java.util.Scanner;

class HelloWorld{

   public static int[] getMaxDiffPair(int[] A) {
      int n = A.length;
      if (n < 2) {
         return new int[]{-1, -1};
      }
      int[] maxDiffPair = {-1, -1};
      int maxDiff = Integer.MIN_VALUE;

      for (int i = 0; i < n - 1; i++) {
         for (int j = i + 1; j < n; j++) {
            if (A[j] > A[i] && A[j] - A[i] > maxDiff) {
               maxDiff = A[j] - A[i];
               maxDiffPair[0] = A[i];
               maxDiffPair[1] = A[j];
            }
         }
      }
      return maxDiffPair;
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the size of the array: ");
      int n = scanner.nextInt();
      int[] A = new int[n];
      System.out.println("Enter the elements of the array:");
      for (int i = 0; i < n; i++) {
         A[i] = scanner.nextInt();
      }
      int[] maxDiffPair = getMaxDiffPair(A);

      if (maxDiffPair[0] != -1) {
         System.out.println("The maximum difference is " + (maxDiffPair[1] - maxDiffPair[0]));
         System.out.println("The pair is (" + maxDiffPair[0] + ", " + maxDiffPair[1] + ")");
      } else {
         System.out.println("No valid pair found to calculate the maximum difference.");
      }
      scanner.close();
   }
}
```

Question:5

Given an array of integers of size N, the task is to find the first non-repeating element in this array.
Examples:
Input: {-1, 2, -1, 3, 0}
Output: 2
Explanation: The first number that does not repeat is : 2
Input: {9, 4, 9, 6, 7, 4}
Output: 6

Answer:-

```
import java.util.Scanner;

public class HelloWorld {

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the size of the array: ");
      int n = scanner.nextInt();
      int[] arr = new int[n];
      System.out.println("Enter the elements of the array:");
      for (int i = 0; i < n; i++) {
         arr[i] = scanner.nextInt();
      }
      for (int i = 0; i < n; i++) {
         boolean isRepeating = false;

         for (int j = 0; j < n; j++) {
            if (i != j && arr[i] == arr[j]) {
               isRepeating = true;
               break;
            }
         }
         if (!isRepeating) {
            System.out.println("The first non-repeating element is: " + arr[i]);
            scanner.close();
            return;
         }
      }
      System.out.println("No non-repeating element found in the array.");
      scanner.close();
   }
}
```

Question:6

Minimize the maximum difference between the heights
Given the heights of N towers and a value of K, Either increase or decrease the height of every tower by K (only once) where K > 0. After modifications, the task is to minimize the difference between the heights of the longest and the shortest tower and output its difference.
Examples:
Input: arr[] = {1, 15, 10}, k = 6 Output: Maximum difference is 5.

Answer:-

```
 import java.util.*;
public class HelloWorld {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
                  System.out.print("Enter the size of the array: ");
      int n = scanner.nextInt();
              int[] arr = new int[n];
      System.out.println("Enter the elements of the array:");
      for (int i = 0; i < n; i++) arr[i] = scanner.nextInt();
      System.out.print("Enter the value of K: ");
      int k = scanner.nextInt();
      int ans = getMinDiff(arr, n, k);
      System.out.println("Minimum maximum difference after modifications is: " + ans);
      scanner.close();
   }
   public static int getMinDiff(int[] arr, int n, int k) {
      Arrays.sort(arr);
      int ans = arr[n - 1] - arr[0];
      for (int i = 1; i < n; i++) {
         if (arr[i] - k >= 0) {
            int tempMin = Math.min(arr[0] + k, arr[i] - k);
            int tempMax = Math.max(arr[i - 1] + k, arr[n - 1] - k);
            ans = Math.min(ans, tempMax - tempMin);
         }
      }
      return ans;
   }
}
```