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)

//Code

```java
import java.util.Scanner;

public class FindSumPair {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       int n = sc.nextInt();
       int[] arr = new int[n];

       for(int i = 0; i < n; i++){
           arr[i] = sc.nextInt();
       }
       int sum = sc.nextInt();

       findPairSum(arr,sum);
   }
   static void findPairSum(int[] arr, int sum){

       int flag = 0;
       for(int i = 0; i < arr.length; i++){
           for(int j = i+1; j < arr.length; j++){
               if(arr[i] + arr[j] == sum){
                   flag = 1;
                   System.out.println("Pair found ("+
arr[i] + " ,"+ arr[j]+")");
                   break;
               }
           }
           if(flag == 1){
               break;
           }
       }
   }
}
```

```
// OUTPUT
//     6
//     8 7 2 5 3 1
//     10
//     Pair found (8 ,2)
```

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 }

```java
import java.util.Scanner;

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

       int n = sc.nextInt();
       int [] arr = new int[n];

       for(int i = 0; i < n; i++){
           arr[i] = sc.nextInt();
       }

       replaceWithProduct(arr, n);
   }
   static void replaceWithProduct(int[] arr, int n){
       int []left_pro =  new int[n];
       int [] right_pro = new int[n];

       int[] ans = new int[n];

       int left = 1;
       int right = 1;

       for(int i = 0; i < n; i++){
           left_pro[i] = left;
           left *= arr[i];
```

```
            right_pro[n-1-i] = right;
            right *= arr[n-1-i];

        }

        for(int i = 0; i < n; i++){
            ans[i] = left_pro[i] * right_pro[i];
        }

        for(int i = 0; i < n; i++){
            System.out.print(ans[i] + " ");
        }
    }
}
//OUTPUT
//          5
//          1 2 3 4 5
//          120 60 40 30 24
```

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.

```
import java.util.Scanner;

public class CircularSubarraySum {
   public static void main(String[] args) {

       Scanner sc = new Scanner(System.in);
       int n = sc.nextInt();
       int[] arr = new int[n];

       for(int i = 0; i < n; i++){
           arr[i] = sc.nextInt();
```

```java
        }

        maxCircularSum(arr, n);
    }

    static void maxCircularSum(int[] arr, int n){
        int res = arr[0];

        for(int i = 0; i < n; i++){
            int curr_max = arr[i];
            int curr_sum = arr[i];

            for(int j = 1; j < n; j++){
                int idx = (i+j) % n;
                curr_sum += arr[idx];
                curr_sum = Math.max(curr_max, curr_sum);
            }
            res = Math.max(res, curr_sum);
        }
        System.out.println("Max sum found " + res);
    }
}
//OUTPUT
//9
//2 1 -5 4 -3 1 -3 4 -1
//Max sum found 6
```

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)

```java
import java.util.Scanner;

public class MaxDifference {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] arr = new int[n];

        for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
        }

        findMaxDiff(arr, n);
    }
    static void findMaxDiff(int[] arr, int n){
        int maxDiff = arr[1]- arr[0];
        int minVal = arr[0];

        for(int j = 1; j < n; j++){
            int curr_max = arr[j] - minVal;
            maxDiff = Math.max(maxDiff,curr_max);
            minVal = Math.min(minVal, arr[j]);

        }
        System.out.println("The maximum difference is "
+ maxDiff );

    }
}
//OUTPUT
//   7
//   2 7 9 5 1 3 5
//   The maximum difference is 7
```

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

```java
import java.util.Scanner;
```

```java
public class FirstNonRepeating {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       int n = sc.nextInt();
       int[] arr = new int[n];

       for(int i = 0; i < n; i++){
           arr[i] = sc.nextInt();
       }

       System.out.println(findFirstNonRepeating(arr,
n));
   }
   static int findFirstNonRepeating(int[] arr, int n){
       for(int i = 0; i < n; i++){
           boolean isUnique = true;
           for(int j = 0; j < n; j++){
               if(i != j && arr[i] == arr[j]){
                   isUnique = false;
                   break;
               }
           }
           if(isUnique == true){
              return arr[i];
           }
       }
       return -1;
   }
}
//OUTPUT
// 5
// -1 2 -1 3 0
// 2
```

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.
Explanation: Change 1 to 7, 15 to 9 and 10 to 4. Maximum difference is 5 (between 4 and 9). We can't get a lower difference.
Input: arr[] = {1, 5, 15, 10}, k = 3
Output: Maximum difference is 8, arr[] = {4, 8, 12, 7}

```java
import java.util.Arrays;
import java.util.Scanner;

public class MinimizeHeight {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       int n = sc.nextInt();
       int[] arr = new int[n];

       for(int i = 0; i < n; i++){
           arr[i] = sc.nextInt();
       }
       int k = sc.nextInt();

       findMinHeight(arr, n, k);
   }

   static void findMinHeight(int[] arr, int n, int k){
       Arrays.sort(arr);

       int max = 0, min = 0;
       int ans = arr[n-1] - arr[0];
       int lar = arr[n-1] - k, sml = arr[0] + k;

       for(int i = 0; i < n - 1; i++){
           min = Math.min(sml, arr[i+1] - k);
           max = Math.max(lar, arr[i] + k);
           if(min < 0)
               continue;
           ans = Math.min(ans, max - min);
       }
```

```
        System.out.println("Maximum difference is " +
ans);

    }
}
//OUTPUT
// 4
// 1 5 15 10
// 3
// Maximum difference is 8
```