WEEK – 1:

```
/*      Ques 1. Given an array of non-negative integers, design a linear
        algorithm and implement it using a program to find whether given key
        element is present in the array or not. Also, find total number of
        comparisons for each input case.
        (Time Complexity = O(n), where n is the size of input)
*/

import java.util.*;
public class Week1 {

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

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                           " + t + ":");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                           t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }

         // Input for key element to search for
         System.out.print("Enter the key element to search for in test
                           case " + t + ": ");
         int key = sc.nextInt();
         int comparisons = 0;
         boolean found = false;
         for(int i = 0; i < n; i++){
            comparisons++;
            if(arr[i] == key){
               found = true;
               break;
            }
         }
         if(found){
            System.out.println("Key element " + key + " is present is
                                 array for test case " + t);
```

```
        }else{
            System.out.println("Key element " + key + " is not
                                present in array for test case " + t);
        }

        // Output total number of comparisons made
        System.out.println("Total number of comparisons made for test
                            case " + t + ": " + comparisons);
    }
  }

}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week1.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week1

Enter the size of the array for test case1:5
Enter the array elements for test case 1: 2 4 6 8 10
Enter the key element to search for in test case 1: 6
Key element 6 is present is array for test case 1
Total number of comparisons made for test case 1: 3

Enter the size of the array for test case2:7
Enter the array elements for test case 2: 12 23 34 45 56 67 78
Enter the key element to search for in test case 2: 100
Key element 100 is not present in array for test case 2
Total number of comparisons made for test case 2: 7
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

```
/*      Ques 2. Given an already sorted array of positive integers, design
        An algorithm and implement it using a program to find whether given
        key element is present in the array or not. Also, find total number
        of comparisons for each input case. (Time Complexity = O(nlogn),
        where n is the size of input).
*/

import java.util.*;
public class Week1 {
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                           " + t + ":");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                           t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }

         // Input for key element to search for
         System.out.print("Enter the key element to search for: ");
         int key = sc.nextInt();

         // Binary search to find whether key element is present in
                           array or not
         int comparisons = 0;
         int left = 0;
         int right = n - 1;
         boolean found = false;
         while(left <= right){
            int mid = (left + right) / 2;
            comparisons++;
            if(arr[mid] == key){
               found = true;
               break;
            }else if(arr[mid] < key){
```

```
                left = mid + 1;
            }else{
                right = mid - 1;
            }
        }
        if(found){
            System.out.print("Present ");
        }else{
            System.out.print("Not Presentm ");
        }
        System.out.print(comparisons);

    }

}
}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week1.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week1
Enter the number of test cases: 2

Enter the size of the array for test case 1:5
Enter the array elements for test case 1: 2 4 6 8 10
Enter the key element to search for: 8
Present 2
Enter the size of the array for test case 2:7
Enter the array elements for test case 2: 12 20 30 35 40 45 98
Enter the key element to search for: 100
Not Presentm 3
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

```
/*      Ques 3.  Given an already sorted array of positive integers, design an algorithm
        and implement it using a program to find whether a given key element is present
        in the sorted array or not. For an array arr[n], search at the indexes arr[0], arr[2],
        arr[4],.....,
        arr[2k] and so on. Once the interval (arr[2k] < key < arr[ 2k+1] ) is found,
        perform a linear search operation from the index 2k to find the element key.
        (Complexity < O(n), where n is the number of elements need to be scanned for
        searching): Jump Search
*/

import java.util.*;
public class Week1 {

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

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                           " + t + ": ");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                           t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }

         // Input for key element to search for
         System.out.print("Enter the key element to search for: ");
         int key = sc.nextInt();

                  // Jump search to find whether key element is present in array or not
         int comparisons = 0;
         int step = (int) Math.floor(Math.sqrt(n));
         int prev = 0;
         while(arr[Math.min(step, n) - 1] < key){
            comparisons++;
            prev = step;
            step += (int)Math.floor(Math.sqrt(n));
            if(prev >= n){
```

```
                break;
            }
        }
        while(arr[prev] < key){
            comparisons++;
            prev++;
            if(prev == Math.min(step, n)){
                break;
            }
        }
        if(arr[prev] == key){
            System.out.print("Present");
        }else{
            System.out.print("Not Present");
        }
        System.out.println(comparisons);

    }
  }
}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week1.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week1
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 2 4 6 8 10
Enter the key element to search for: 10
Present2

Enter the size of the array for test case 2: 7
Enter the array elements for test case 2: 12 23 34 45 56 67 78
Enter the key element to search for: 45
Present2
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

WEEK – 2:

```
/*      Ques 1. Given a sorted array of positive integers containing few duplicate
        elements, design an algorithm and implement it using a program to find whether
        the given key element is present in the array or not. If present, then also find the
        number of copies of given key. (Time Complexity = O(log n))
*/


import java.util.*;
public class Week2{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                           " + t + ": ");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                           t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }

         // Input for key element to search for
         System.out.print("Enter the key element to search for: ");
         int key = sc.nextInt();
         int first = 0;
         int last = n - 1;
         int mid;
         int count = 0;
         while(first <= last){
            mid = (first + last) / 2;
            if(arr[mid] == key){
               count++;
               int left = mid - 1;
               while(left >= 0 && arr[left] == key){
                  count++;
                  left--;
               }
```

```
                int right = mid + 1;
                while(right < n && arr[right] == key){
                    count++;
                    right++;
                }
                break;
            }else if(arr[mid] < key){
                first = mid + 1;
            }else{
                last = mid - 1;
            }
        }
        if(count == 0){
            System.out.print("Key not Present");
        }else{
            System.out.print(key);
            System.out.print(" " + count);
        }
    }
  }
}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week2.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week2
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 2 4 6 8 10
Enter the key element to search for: 8
8 1
Enter the size of the array for test case 2: 7
Enter the array elements for test case 2: 23 43 45 55 65 76 98
Enter the key element to search for: 100
Key not Present
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

```
/*      Ques 2. Given a sorted array of positive integers, design an algorithm and
        implement it using a program to find three indices i, j, k such that arr[i] + arr[j] =
        arr[k].
*/


import java.util.*;
public class Week2{

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

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                              " + t + ": ");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                              t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }

                  // Find three indices i, j, k such that arr[i] + arr[j] = arr[k]
         boolean found = false;
         for(int i = 0; i < n - 2; i++){
            int j = i + 1;
            int k = n - 1;
            while(j < k){
               if(arr[i] + arr[j] == arr[k]){
                  found = true;
                  System.out.println(i + ", " + j + ", " + k);
                  break;
               }else if (arr[i] + arr[j] < arr[k]){
                  j++;
               }else{
                  k--;
               }
            }
            if(found){
```

```
                break;
            }
        }

        if(!found){
            System.out.println("No Sequence Found");
        }
    }
  }

}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week2.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week2
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 2 4 6 8 10
0, 3, 4

Enter the size of the array for test case 2: 7
Enter the array elements for test case 2: 23 45 67 87 89 90 99
No Sequence Found
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

```
/*      Ques 3. Given an array of nonnegative integers, design an algorithm and a
        program to count the number of pairs of integers such that their difference is
        equal to a given key, K.
*/


import java.util.*;
public class Week2{

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

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                           " + t + ": ");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                           t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }

         // Input for key K
         System.out.print("Enter the key: ");
         int K = sc.nextInt();

         // Count pairs whose difference is K
         int count = 0;
         for(int i = 0; i < n - 1; i++){
            for(int j = i + 1; j < n; j++){
               if(Math.abs(arr[i] - arr[j]) == K){
                  count++;
               }
            }
         }
         System.out.println(count);
      }
   }

}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week2.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week2
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 2 4 6 8 10
Enter the key: 10
0

Enter the size of the array for test case 2: 7
Enter the array elements for test case 2: 12 34 44 54 56 67 87
Enter the key: 10
2
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

WEEK – 3:

```
/*      Ques 1. Given an unsorted array of integers, design an algorithm and a program to
        sort the array using insertion sort. Your program should be able to find number of
        comparisons and shifts ( shifts -
        total number of times the array elements are shifted from their place) required for
        sorting the array.
*/

import java.util.*;
public class Week3{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                           " + t + ": ");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                                    t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }

                  // Sort the array using insertion sort and count comparisons and shifts
         int comparisons = 0;
         int shifts = 0;
         for(int i = 1; i < n; i++){
            int key = arr[i];
            int j = i - 1;
            while(j >= 0 && arr[j] > key){
               arr[j + 1] = arr[j];
               j--;
               comparisons++;
               shifts++;
            }
            arr[j + 1] = key;
            shifts++;
         }
         System.out.print("Sorted array: ");
```

```
            for(int i = 0; i < n; i++){
                System.out.print(arr[i] + " ");
            }
            System.out.println();
            System.out.println("Number of comparisons: " + comparisons);
            System.out.println("Number of shifts: " + shifts);

        }
    }
}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week3.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week3
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 2 4 6 8 10
Sorted array: 2 4 6 8 10
Number of comparisons: 0
Number of shifts: 4

Enter the size of the array for test case 2: 7
Enter the array elements for test case 2: 87 45 12 98 -7 -56 0
Sorted array: -56 -7 0 12 45 87 98
Number of comparisons: 16
Number of shifts: 22
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

```
/*      Ques 2. Given an unsorted array of integers, design an algorithm and implement a
        program to sort this array using selection sort. Your program should also find
        number of comparisons and number of
        swaps required.
*/

import java.util.*;
public class Week3{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                           " + t + ": ");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                           t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }

                  // Sort the array using selection sort and count comparisons and swaps
         int comparisons = 0;
         int swaps = 0;
         for(int i = 0; i < n - 1; i++){
            int minIndex = i;
            for(int j = i + 1; j < n; j++){
               if(arr[j] < arr[minIndex]){
                  minIndex = j;
               }
               comparisons++;
            }
            if(minIndex != i){
               int temp = arr[i];
               arr[i] = arr[minIndex];
               arr[minIndex] = temp;
               swaps++;
            }
         }
```

```
            System.out.print("Sorted array: ");
            for(int i = 0; i < n; i++){
                System.out.print(arr[i] + " ");
            }
            System.out.println();
            System.out.println("Number of comparisons: " + comparisons);
            System.out.println("Number of swaps: " + swaps);
        }
    }
}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week3.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week3
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 23 43 1 2 -9
Sorted array: -9 1 2 23 43
Number of comparisons: 10
Number of swaps: 4

Enter the size of the array for test case 2: 7
Enter the array elements for test case 2: 12 98 45 -9 -45 0 -99
Sorted array: -99 -45 -9 0 12 45 98
Number of comparisons: 21
Number of swaps: 5
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

```
/*      Ques 3. Given an unsorted array of positive integers, design
        an algorithm and implement it using a program to find whether there are any
        duplicate elements in the array or not. (use sorting) (Time
        Complexity = O(n log n))
*/


import java.util.*;
public class Week3{

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

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                           " + t + ": ");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                           t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }
         Arrays.sort(arr);
         int flag = 0;
         for(int i = 0; i < n-1 ; i++){
            if(arr[i] == arr[i + 1]){
               flag = 1;
               break;
            }
         }
         if(flag == 1){
            System.out.println("Duplicate elements found");
         }else{
            System.out.println("No duplicate elements found");
         }
      }
   }
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week3.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week3
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 10 8 6 4 2
No duplicate elements found

Enter the size of the array for test case 2: 7
Enter the array elements for test case 2: 12 12 34 45 58 65 2 65
Duplicate elements found
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

WEEK – 4:

```
/*      Ques 1. Given an unsorted array of integers, design an algorithm and implement it
        using a program to sort an array of elements by dividing the array into two
        subarrays and combining these subarrays
        after sorting each one of them. Your program should also find number of
        comparisons and inversions during sorting the array.
*/


import java.util.*;
public class Week4{

   static int countComparisons, countInversions;
   public static void merge(int[] arr, int left, int mid, int right){
      int n1 = mid - left + 1;
      int n2 = right - mid;

      int[] L = new int[n1];
      int[] R = new int[n2];

      for(int i = 0; i < n1; i++)
         L[i] = arr[left + i];
      for(int j = 0; j < n2; j++)
         R[j] = arr[mid + 1 + j];

      int i = 0, j = 0;
      int k = left;
      while(i < n1 && j < n2){
         countComparisons++;
         if(L[i] <= R[j]){
            arr[k] = L[i];
            i++;
         }else{
            arr[k] = R[j];
            j++;
            countInversions += (n1 - i);
         }
         k++;
      }
      while(i < n1){
         arr[k] = L[i];
         i++;
         k++;
      }
      while(j < n2){
         arr[k] = R[j];
         j++;
         k++;
```

```
        }
    }

    public static void mergeSort(int[] arr, int left, int right){
        if(left < right){
            int mid = (left + right) / 2;
            mergeSort(arr, left, mid);
            mergeSort(arr, mid + 1, right);
            merge(arr, left, mid, right);
        }
    }

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

        // Input for number of test cases
        System.out.print("Enter the number of test cases: ");
        int T = sc.nextInt();

        // Loop for each test case
        for(int t = 1; t <= T; t++){
            // Input for size of array
            System.out.print("\nEnter the size of the array for test case
                              " + t + ": ");
            int n = sc.nextInt();

            // Input for array elements
            int[] arr = new int[n];
            System.out.print("Enter the array elements for test case " +
                              t + ": ");
            for(int i = 0; i < n; i++){
                arr[i] = sc.nextInt();
            }
            int l = arr.length;

            mergeSort(arr, 0, n - 1);

            System.out.println("Sorted array:");
            for(int i = 0; i < l; i++)
                System.out.print(arr[i] + " ");

            System.out.println("\nNumber of comparisons: " +
                              countComparisons);
            System.out.println("Number of inversions: " +
                              countInversions);
        }
    }
}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week4.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week4
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 98 23 45 -9 0
Sorted array:
-9 0 23 45 98
Number of comparisons: 6
Number of inversions: 8

Enter the size of the array for test case 2: 7
Enter the array elements for test case 2: -23 -9 0 -45 23 45 4
Sorted array:
-45 -23 -9 0 4 23 45
Number of comparisons: 17
Number of inversions: 13
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

```
/*      Ques 2. Given an unsorted array of integers, design an algorithm and
        implement it using a program to sort an array of elements by partitioning the array
        into two subarrays based on a pivot element
        such that one of the sub array holds values smaller than the pivot element while
        another sub array holds values greater than the pivot element. Pivot element should
        be selected randomly from the array. Your program should also find number of
        comparisons and swaps required for sorting the array.
*/


import java.util.*;
public class Week4{

   static int comparisons = 0;
   static int swaps = 0;
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                           " + t + ": ");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                           t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }
         quickSort(arr, 0, arr.length-1);
         System.out.println("Sorted array: ");
         for(int i = 0; i < arr.length; i++){
            System.out.print(arr[i] + " ");
         }
         System.out.println("\nNumber of comparisons: " +comparisons);
         System.out.println("Number of swaps: " + swaps);
      }
   }
   public static void quickSort(int[] arr, int left, int right){
      if(left < right){
```

```
            int pivIdx = partition(arr, left, right);
            quickSort(arr, left, pivIdx-1);
            quickSort(arr, pivIdx+1, right);
        }
    }
    public static int partition(int[] arr, int left, int right){
        int pivotIndex = left + (int)(Math.random() * (right-left + 1));
        int pivot = arr[pivotIndex];
        swap(arr, pivotIndex, right); // move pivot to end of array
        int i = left-1;
        for(int j = left; j < right; j++){
            if(arr[j] < pivot){
                i++;
                swap(arr, i, j);
            }
            comparisons++;
        }
        swap(arr, i+1, right); // move pivot to its final position
        swaps++;
        return i+1;
    }
    public static void swap(int[] arr, int i, int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
        swaps++;
    }
}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week4.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week4
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 9 7 5 23 0
Sorted array:
0 5 7 9 23
Number of comparisons: 8
Number of swaps: 11

Enter the size of the array for test case 2: 7
Enter the array elements for test case 2: -32 0 -9 5 12 43 87
Sorted array:
-32 -9 0 5 12 43 87
Number of comparisons: 21
Number of swaps: 28
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

```
/*      Ques 3. Given an unsorted array of integers, design an algorithm and Implement
        it using a program to find Kth smallest or largest element in the array. (Worst case
        Time Complexity = O(n))
*/

import java.util.*;
public class Week4{

    public static int quickselect(int[] arr, int k){
        int left = 0, right = arr.length - 1;
        while(left <= right){
            int pivotIndex = partition(arr, left, right);
            if(pivotIndex == k - 1){
                return arr[pivotIndex];
            }else if(pivotIndex < k - 1){
                left = pivotIndex + 1;
            }else{
                right = pivotIndex - 1;
            }
        }
        return -1;
    }

    public static int partition(int[] arr, int left, int right) {
        int pivot = arr[right];
        int i = left - 1;
        for(int j = left; j <= right - 1; j++){
            if(arr[j] <= pivot){
                i++;
                swap(arr, i, j);
            }
        }
        swap(arr, i + 1, right);
        return i + 1;
    }

    public static void swap(int[] arr, int i, int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

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

        // Input for number of test cases
        System.out.print("Enter the number of test cases: ");
        int T = sc.nextInt();
```

```
        // Loop for each test case
        for(int t = 1; t <= T; t++){
            // Input for size of array
            System.out.print("\nEnter the size of the array for test case
                                " + t + ": ");
            int n = sc.nextInt();

            // Input for array elements
            int[] arr = new int[n];
            System.out.print("Enter the array elements for test case " +
                                t + ": ");
            for(int i = 0; i < n; i++){
                arr[i] = sc.nextInt();
            }
            System.out.print("Enter the value of K to find Kth smallest
                                element: ");
            int k = sc.nextInt();
            int kthSmallest = quickselect(arr, k);
            System.out.println(kthSmallest);
        }
    }
}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week4.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week4
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 9 7 5 23 1
Enter the value of K to find Kth smallest element: 4
9

Enter the size of the array for test case 2: 7
Enter the array elements for test case 2: 123 543 678 456 158 458 789
Enter the value of K to find Kth smallest element: 158
-1
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

WEEK – 5:

```
/*      Ques 1. Given an unsorted array of alphabets containing duplicate elements.
        Design an algorithm and implement it using a program to find which alphabet has
        maximum number of occurrences and print it. (Time Complexity = O(n)) (Hint:
        Use counting sort)
*/

import java.util.*;
public class Week5{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the number of test cases- ");
      int t = sc.nextInt();
      sc.nextLine();
      while(t-- > 0){
         String s = sc.nextLine();
         int[] count = new int[26];
         for(int i = 0; i < s.length(); i++){
            char c = s.charAt(i);
            count[c - 'a']++;
         }
         int maxCount = 0;
         int maxCountIndex = 0;
         for(int i = 0; i < 26; i++){
            if(count[i] > maxCount){
               maxCount = count[i];
               maxCountIndex = i;
            }
         }
         char maxOccurringAlphabet = (char) (maxCountIndex + 'a');
         System.out.println("Max occurring alphabet: " +
                           maxOccurringAlphabet + " " + maxCount);
      }
   }
}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week5.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week5
Enter the number of test cases-
2
qwertyuirtkjbuygesr
Max occurring alphabet: r 3
qwertyuiopoiuyttretttwtuetttt
Max occurring alphabet: t 11
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

```
/*      Ques 2. Given an unsorted array of integers, design an algorithm and implement it
        using a program to find whether two elements exist such that their sum is equal to
        the given key element. (Time Complexity = O(n log n))
*/


import java.util.*;
public class Week5{

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

      // Input for number of test cases
      System.out.print("Enter the number of test cases: ");
      int T = sc.nextInt();

      // Loop for each test case
      for(int t = 1; t <= T; t++){
         // Input for size of array
         System.out.print("\nEnter the size of the array for test case
                           " + t + ": ");
         int n = sc.nextInt();

         // Input for array elements
         int[] arr = new int[n];
         System.out.print("Enter the array elements for test case " +
                           t + ": ");
         for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
         }System.out.print("Enter the key element: ");
         int key = sc.nextInt();
         Arrays.sort(arr);
         int left = 0, right = n - 1;
         while(left < right){
            int sum = arr[left] + arr[right];
            if(sum == key){
               System.out.println(arr[left] + " " + arr[right]);
               return;
            }else if(sum < key){
               left++;
            }else{
               right--;
            }
         }
         System.out.println("No elements found");
      }
   }
}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week5.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week5
Enter the number of test cases: 2

Enter the size of the array for test case 1: 5
Enter the array elements for test case 1: 2 4 6 8 10
Enter the key element: 10
2 8
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```

```java
/*      Ques 3. You have been given two sorted integer arrays of size m and n. Design an
        algorithm and implement it using a program to find list of elements which are
        common to both. (Time Complexity = O(m+n))
*/
import java.util.*;
public class Week5{

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("\nEnter the number of tese cases: ");
      int T = sc.nextInt();
      for(int t=1;t<=T;t++){

         System.out.print("Enter the size of test case " + t + " array 1: ");
         int m = sc.nextInt();
         int[] arr1 = new int[m];
         System.out.print("Enter the elements of array 1: ");
         for (int i = 0; i < m; i++) {
            arr1[i] = sc.nextInt();
         }
         Arrays.sort(arr1);

         System.out.print("Enter the test case " + t + " size of array 2: ");
         int n = sc.nextInt();
         int[] arr2 = new int[n];
         System.out.print("Enter the elements of array 2: ");
         for (int i = 0; i < n; i++) {
            arr2[i] = sc.nextInt();
         }
         Arrays.sort(arr2);

         int i = 0, j = 0;
         System.out.print("Common elements: ");
         while (i < m && j < n) {
            if (arr1[i] == arr2[j]) {
               System.out.print(arr1[i] + " ");
               i++;
               j++;
            } else if (arr1[i] < arr2[j]) {
               i++;
            } else {
               j++;
            }
         }
      }
   }

}
```

**********OUTPUT********

```
PS C:\Users\deepa\OneDrive\Desktop\DAA> javac Week5.java
PS C:\Users\deepa\OneDrive\Desktop\DAA> java Week5

Enter the number of tese cases: 2
Enter the size of test case 1 array 1: 5
Enter the elements of array 1: 2 5 34 1 6
Enter the test case 1 size of array 2: 7
Enter the elements of array 2: 2 45 67 34 1 98 10
Common elements: 1 2 34 Enter the size of test case 2 array 1: 7
Enter the elements of array 1: 1 2 3 4 5 6 7
Enter the test case 2 size of array 2: 7
Enter the elements of array 2: 1 2 3 4 5 6 7
Common elements: 1 2 3 4 5 6 7
PS C:\Users\deepa\OneDrive\Desktop\DAA>
```