DAY TWO OF WWC CAMP:

1.Selection sort.

Code:-

```
import java.util.*;
public class selectionSort
{
   public static void main(String[] args)
   {
      int arr[] = inputArray();
      printMethod(arr);
      System.out.println("your sorted array is:-");
      sorting(arr);

   }
   public static int[] inputArray()
   {
      System.out.println("WE ARE IN INPUT STAGE:");
      Scanner sc = new Scanner(System.in);
      System.out.println("enter size of your array:-");
      int size = sc.nextInt();
      int arr[] = new int[size];

      System.out.println("enter value " +(size)+ " of your array");
      for(int i = 0; i<size; i++)
      {
         System.out.print("enter " + (i) + ":-" + " ");
         arr[i] = sc.nextInt();
      }
      return arr;
```

```
}
public static void printMethod(int arr[])
{
   System.out.println("WE ARE ON PRINTING STAGE:");
   System.out.println("your array is:-");
   for(int i=0; i<arr.length; i++)
   {
      System.out.println(arr[i] +" ");
   }

}
public static void sorting(int arr[])
{
   System.out.println("WE ARE IN SORTING STAGE:-");
   System.out.println("your sorted array is:");
   for(int i=0; i<arr.length-1; i++)
   {
      int minIndex = i;
      for(int j=i+1; j<arr.length; j++)
      {
         if(arr[j]<arr[minIndex])
         {
            minIndex = j;
         }
      }
      int temp = arr[minIndex];
      arr[minIndex] = arr[i];
      arr[i] = temp;

   System.out.print("steps" +(i+1)+":-");
   System.out.println(Arrays.toString(arr));
```

```
    }


  }
}
```

2.Bubble sort.

Code:-

```
import java.util.*;
public class bubbleSort
{
        public static void main(String[] args) {
           int arr[] = inputArray();
           printMethod(arr);
           System.out.println("Your sorted array procedure:");
           bubbleSort(arr);
           //System.out.print(Arrays.toString(arr));



        }
        public static int[] inputArray()
        {
           System.out.println("WE ARE NOW TAKING YOUR INPUT: ");
           Scanner sc = new Scanner(System.in);
           System.out.println("enter the size of Array:-");
           int size = sc.nextInt();
           int arr[] = new int[size];
           System.out.println("enter " + size + " numbers:-");
           for(int i = 0; i<size; i++)
```

```
    {
        System.out.print("enter " + ( i ) + " :- " + " ");
        arr[i] = sc.nextInt();
    }
    return arr;
}
public static void printMethod(int arr[])
{
    System.out.println("NOW PRINTING YOUR ARRAY:");
    System.out.println("your filled array elements are:-");
    for(int i=0; i<arr.length; i++)
    {
        System.out.println(arr[i] +" ");
    }

}
public static void bubbleSort(int arr[]){
    for(int i = 0; i<arr.length-1; i++)
    {
        for(int j=0; j<arr.length-1; j++)
        {
            if(arr[j]>arr[j+1])
            {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
        System.out.println("Step " +(i+1) );
         System.out.println(Arrays.toString(arr));
    }
```

```
        }
}
```

3.Insertion sort.

Code:-

```
import java.util.Arrays;

public class InsertionSort {
    public static void main(String[] args) {
        int[] arr = {29, 10, 14, 37, 13};

        System.out.println("Original array: " + Arrays.toString(arr));
        insertionSort(arr);
        System.out.println("Sorted array: " + Arrays.toString(arr));
    }

    public static void insertionSort(int[] arr) {
        for (int i = 1; i < arr.length; i++) {
            int key = arr[i];
            int j = i - 1;

            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j--;
            }

            arr[j + 1] = key;

            System.out.println("Step " + i + ": " + Arrays.toString(arr));
        }
```

```
    }
}
```

4.Quick sort.

Code:-

```
import java.util.Arrays;

public class QuickSort {
    public static void main(String[] args) {
        int[] arr = {29, 10, 14, 37, 13};

        System.out.println("Original array: " + Arrays.toString(arr));
        quickSort(arr, 0, arr.length - 1);
        System.out.println("Sorted array: " + Arrays.toString(arr));
    }

    public static void quickSort(int[] arr, int low, int high) {
        if (low < high) {
            int pi = partition(arr, low, high);



            quickSort(arr, low, pi - 1);
            quickSort(arr, pi + 1, high);
        }
    }

    public static int partition(int[] arr, int low, int high) {
        int pivot = arr[high];
        int i = low - 1;
```

```
        for (int j = low; j < high; j++) {
            if (arr[j] < pivot) {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }


        int temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;

        return i + 1;  // Return the partition index
    }
}
```

5.Merge sort.

Code:-

```
import java.util.Arrays;


public class MergeSort {
    public static void main(String[] args) {
        int[] arr = {29, 10, 14, 37, 13};

        System.out.println("Original array: " + Arrays.toString(arr));
        mergeSort(arr, 0, arr.length - 1);
        System.out.println("Sorted array: " + Arrays.toString(arr));
    }
```

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

        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);



        merge(arr, left, mid, right);
    }
}

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) {
```

```
            if (L[i] <= R[j]) {
                arr[k] = L[i];
                i++;
            } else {
                arr[k] = R[j];
                j++;
            }
            k++;
        }

        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }



        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
    }
}
```

6.Bucket sort.

Code:-

```
import java.util.ArrayList;
import java.util.Collections;
```

```
public class BucketSort {
    public static void main(String[] args) {
        double[] arr = {0.42, 0.32, 0.23, 0.52, 0.25, 0.47, 0.51};

        System.out.println("Original array:");
        for (double num : arr) {
            System.out.print(num + " ");
        }

        bucketSort(arr);

        System.out.println("\nSorted array:");
        for (double num : arr) {
            System.out.print(num + " ");
        }
    }

    public static void bucketSort(double[] arr) {
        int n = arr.length;

        ArrayList<Double>[] buckets = new ArrayList[n];
        for (int i = 0; i < n; i++) {
            buckets[i] = new ArrayList<>();
        }

        for (double num : arr) {
            int bucketIndex = (int) (num * n); // Bucket index
            buckets[bucketIndex].add(num);
        }

        for (ArrayList<Double> bucket : buckets) {
```

```
            Collections.sort(bucket);
        }

        int index = 0;
        for (ArrayList<Double> bucket : buckets) {
            for (double num : bucket) {
                arr[index++] = num;
            }
        }
    }
}
```