CSW 2

ASSIGNMENT 3

2.
```
public class Sort012 {
    public static void sort012(int[] arr) {
        int low = 0;
        int mid = 0;
        int high = arr.length - 1;

        while (mid <= high) {
            switch (arr[mid]) {
                case 0:
                    swap(arr, low, mid);
                    low++;
                    mid++;
                    break;
                case 1:
                    mid++;
                    break;
                case 2:
```

```
                swap(arr, mid, high);
                high--;
                break;
            }
        }
    }

    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) {
        int[] arr = {0, 1, 2, 0, 1, 2};
        System.out.println("Before sorting: " +
Arrays.toString(arr));

        sort012(arr);
        System.out.println("After sorting: " +
Arrays.toString(arr));
    }
}
```

3.
```
public class MinSwaps {
    public static int minSwaps(int[] arr, int val) {
        int count = 0;
        int numLessThanVal = 0;

        // Count the number of elements less than val
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] < val) {
                numLessThanVal++;
            }
        }

        // Count the number of elements less than val in the first numLessThanVal elements
        int countLessThanVal = 0;
        for (int i = 0; i < numLessThanVal; i++) {
            if (arr[i] < val) {
                countLessThanVal++;
            }
        }

        // Calculate the minimum number of swaps required
```

```
        int minSwaps = Integer.MAX_VALUE;
        for (int i = numLessThanVal; i < arr.length; i++) {
            if (arr[i] < val) {
                count++;
            }
            if (i - numLessThanVal >= 0 && arr[i - numLessThanVal] < val) {
                count--;
            }
            minSwaps = Math.min(minSwaps, count);
        }

        return minSwaps;
    }

    public static void main(String[] args) {
        int[] arr = {5, 6, 3, 4, 7, 2, 1};
        int val = 5;

        int minSwaps = minSwaps(arr, val);
        System.out.println("Minimum swaps required: " + minSwaps);
    }
```

```
}

4. public class SortArray {
    public static void sortArray(int[] arr1, int[] arr2) {
        Map<Integer, Integer> map = new HashMap<>();

        // Store the index of each element of arr2 in the map
        for (int i = 0; i < arr2.length; i++) {
            map.put(arr2[i], i);
        }

        // Sort arr1 according to the order in arr2 using the
index in the map
        Arrays.sort(arr1, new Comparator<Integer>() {
            public int compare(Integer a, Integer b) {
                if (map.containsKey(a) && map.containsKey(b))
{
                    return map.get(a) - map.get(b);
                } else if (map.containsKey(a)) {
                    return -1;
                } else if (map.containsKey(b)) {
                    return 1;
                } else {
```

```
                return a - b;
            }
        }
    });
}

public static void main(String[] args) {
    int[] arr1 = {5, 4, 3, 2, 1};
    int[] arr2 = {2, 3, 1};

    sortArray(arr1, arr2);

    System.out.println(Arrays.toString(arr1));
}
}
```

5.
```
public class MinimumSwaps {

    public static void main(String[] args) {
        int[] arr = {2, 5, 1, 3, 8, 6, 4, 7};
        int k = 5; // value to compare
        int count = 0;
```

```
// count the number of elements less than k
for (int i = 0; i < arr.length; i++) {
    if (arr[i] < k) {
        count++;
    }
}

// find the number of elements greater than or equal to k within the first count elements
int minSwaps = 0;
for (int i = 0; i < count; i++) {
    if (arr[i] >= k) {
        minSwaps++;
    }
}

int swaps = minSwaps;
for (int i = 0, j = count; j < arr.length; i++, j++) {
    if (arr[i] >= k) {
        swaps--;
    }
```

```
            if (arr[j] >= k) {
                swaps++;
            }
            minSwaps = Math.min(minSwaps, swaps);
        }

        System.out.println("Minimum swaps required: " +
minSwaps);
    }
}
```

6.
```
public class EvenOddSeparator {

    public static void main(String[] args) {
        int[] arr = {2, 5, 1, 3, 8, 6, 4, 7};
        int[] evenArr = new int[arr.length];
        int[] oddArr = new int[arr.length];
        int evenCount = 0;
        int oddCount = 0;

        for (int i = 0; i < arr.length; i++) {
            if (arr[i] % 2 == 0) {
```

```
            evenArr[evenCount] = arr[i];
            evenCount++;
        } else {
            oddArr[oddCount] = arr[i];
            oddCount++;
        }
    }

    System.out.println("Even numbers:");
    for (int i = 0; i < evenCount; i++) {
        System.out.print(evenArr[i] + " ");
    }
    System.out.println();

    System.out.println("Odd numbers:");
    for (int i = 0; i < oddCount; i++) {
        System.out.print(oddArr[i] + " ");
    }
    System.out.println();
  }
}
```

7. import java.util.Arrays;

public class ReductionOperation {

    public static void main(String[] args) {
        int[] arr = {4, 2, 8, 3, 6};
        int n = arr.length;

        Arrays.sort(arr);

        while (n > 0) {
            int min = arr[0];
            System.out.println(n);
            for (int i = 0; i < n; i++) {
                arr[i] -= min;
            }
            Arrays.sort(arr);
            while (n > 0 && arr[n-1] == 0) {
                n--;
            }
        }
    }}

9. import java.util.*;

```
public class ArrayUnionIntersection {
    public static void main(String[] args) {
        int[] arr1 = {2, 5, 7, 8, 9};
        int[] arr2 = {1, 2, 3, 4, 5};

        // finding union
        Set<Integer> unionSet = new HashSet<>();
        for (int i : arr1) {
            unionSet.add(i);
        }
        for (int i : arr2) {
            unionSet.add(i);
        }
        System.out.println("Union of arrays: " + unionSet);

        // finding intersection
        Set<Integer> intersectionSet = new HashSet<>();
        for (int i : arr1) {
            if (Arrays.binarySearch(arr2, i) >= 0) {
                intersectionSet.add(i);
```

```
            }
        }
        System.out.println("Intersection of arrays: " +
intersectionSet);
    }
}
```

10.
```
import java.util.*;

public class IntegerListSorting {
    private List<Integer> list;

    public IntegerListSorting(List<Integer> list) {
        this.list = list;
    }

    public void sortList() {
        Collections.sort(list);
    }

    public int findMin() {
        return Collections.min(list);
```

```
    }

    public int findMax() {
        return Collections.max(list);
    }

    public double findMedian() {
        int size = list.size();
        if (size % 2 == 0) {
            int middleIndex = size / 2;
            int sumOfMiddleElements = list.get(middleIndex -
1) + list.get(middleIndex);
            return (double) sumOfMiddleElements / 2;
        } else {
            int middleIndex = size / 2;
            return (double) list.get(middleIndex);
        }
    }

    public static void main(String[] args) {
        List<Integer> list = new
ArrayList<>(Arrays.asList(7, 2, 9, 4, 1, 5));
```

```
        IntegerListSorting sorter = new
IntegerListSorting(list);

        sorter.sortList();

        System.out.println("Sorted List: " + list);

        System.out.println("Minimum Value: " +
sorter.findMin());

        System.out.println("Maximum Value: " +
sorter.findMax());

        System.out.println("Median Value: " +
sorter.findMedian());

    }
}
```