CSC 151 Assignment #2

1. Honor Code

A. For individual assignments: Jane Doe and John Doe will be replaced by your full name(s)
I affirm that I have carried out my academic endeavors with full academic honesty.
[Signed, Manav Bilakhia]
B. Resources/References
geeksforgeeks

2. Java files and outputs

A. Java files

Class: ArrayOperations

```java
/*
 * I affirm that I have carried out the attached academic endeavors with full academic honesty.
 * Manav Bilakhia (MB)
 */
package assignment;

/**
 * Array operations that use search and sort
 *
 * @Manav Bilakhia
 */

public class ArrayOperations {
    /**
     * doubleExists:
     * Checks if double of an element from the array exists in the same array
     * @param arr an integer array
     * @return true if the array has a double of the other
     * time complexity: O(n^2)
     */

    public static boolean doubleExists(int [] arr)
    {
        if (arr.length != 0) {
            for (int i = 0; i < arr.length; i++) {
                for (int j = 1; j < arr.length - 1; j++) {
                    int m = arr[i];
                    int n = arr[j];
                    if (m == 2 * n)
                        return true;
                    else if (n == 2 * m)
                        return true;
                }
            }
        }
        return false;
    }
    /**
     * moveGivenValue:
     * Given an integer array it moves the integers
     * that are equal to val to the end
     * and maintains the ordering
     *
     * @param arr an integer array
     * @param val a target value
```

```java
     *  time complexity: O(n)
     */
    public static void moveGivenValue(int [] arr, int val)
    {
        int j = 0;
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] != val) {
                int temp = arr[j];
                arr[j] = arr[i];
                arr[i] = temp;
                j++;
            }
        }
    }
    /**
     * sortEvenOdd:
     * Given an integer array it moves the even numbers to the front
     * and odd numbers to the end
     * and it does not maintain the ordering
     *
     * @param arr an integer array
     * @return the arr moving evens to the front
     * time complexity: O(n^2)
     */
    public static int [] sortEvenOdd(int [] arr)
    {
            // Make all odd numbers negative so that they are smaller than all the even
numbers essentially shifting all of them to one side in a simple sort.
            for (int i = 0; i < arr.length; i++)
                if (arr[i] % 2 != 0) // Check for odd y % 2 == 1
                    arr[i] *= -1;

            //sorting the array
            for (int k = 0; k < arr.length; k++)
            {
                for (int j = k + 1; j < arr.length; j++)
                {
                    int tmp = 0;
                    if (arr[k] < arr[j])
                    {
                        tmp = arr[k];
                        arr[k] = arr[j];
                        arr[j] = tmp;
                    }
                }
            }
            // making all odd numbers positive.
            for (int l = 0; l < arr.length; l++)
                if (arr[l] % 2 != 0)
                    arr[l] *= -1;
            return arr;
    }
    /**
     * display:
     * Displays an integer array with a space in between the numbers
     *
     * @param arr an integer array
     * time complexity: O(n)
     */
    public static void display(int [] arr)
    {
        for (int i=0; i <arr.length; i++)
        {
```

```
            if (i!=arr.length-1)
                System.out.print(" "+ arr[i] );
            else
                System.out.print(" "+arr[i]+ "\n");

        }
    }
    /**
     * Tests the operations
     *
     * @param args
     */
    public static void main(String[] args) {
        int[] arr1 = { 9,1,6,3,2,0,1,6 };
        int[] arr2 = { 2,8,6,7,5,2,5,2};
        int[] arr3 = {};
        System.out.println("============================");
        System.out.println("doubleExists tests");
        System.out.println();
        System.out.println("doubleExists(arr1)");
        display(arr1);
        System.out.println(doubleExists(arr1));
        System.out.println();
        System.out.println("doubleExists(arr2)");
        display(arr2);
        System.out.println(doubleExists(arr2));
        System.out.println();
        System.out.println("doubleExists(arr3)");
        display(arr3);
        System.out.println(doubleExists(arr3));
        System.out.println();
        System.out.println("============================");
        int val = 0;
        int[] arr4 = { 2,0,0,2,3,0,0,3 };
        int[] arr5 = { 0 };
        int[] arr6 = { 1,9,7,7 };
        System.out.println("moveGivenValue tests");
        System.out.println();
        System.out.println("moveGivenValue(arr4, 0)");
        System.out.print("Before:");
        display(arr4);
        moveGivenValue(arr4, val);
        System.out.print("After:");
        display(arr4);
        System.out.println();
        System.out.println("moveGivenValue(arr5, 0)");
        System.out.print("Before:");
        display(arr5);
        moveGivenValue(arr5, val);
        System.out.print("After:");
        display(arr5);
        System.out.println();
        System.out.println("moveGivenValue(arr6, 0)");
        System.out.print("Before:");
        display(arr6);
        moveGivenValue(arr6, val);
        System.out.print("After:");
        display(arr6);
        System.out.println();
        val = 1;
        int[] arr7 = { 1, 8,4,5,8,5,1 };
        System.out.println("moveGivenValue(arr7, 1)");
        System.out.print("Before:");
```

```
        display(arr7);
        moveGivenValue(arr7, val);
        System.out.print("After:");
        display(arr7);
        System.out.println();
        System.out.println("============================");
        int[] arr8 = { 6,6,7,6,5,1,6 };
        System.out.println("sortEvenOdd tests");
        System.out.println();
        System.out.println("sortEvenOdd(arr8)");
        System.out.print("Before:");
        display(arr8);
        sortEvenOdd(arr8);
        System.out.print("After:");
        display(arr8);
        System.out.println();
        int[] arr9 = { 3, 1 };
        System.out.println("sortEvenOdd(arr9)");
        System.out.print("Before:");
        display(arr9);
        sortEvenOdd(arr9);
        System.out.print("After:");
        display(arr9);
        System.out.println();
        int[] arr10 = { 8, 6, 2, 4 };
        System.out.println("sortEvenOdd(arr10)");
        System.out.print("Before:");
        display(arr10);
        sortEvenOdd(arr10);
        System.out.print("After:");
        display(arr10);
        System.out.println();
        System.out.println("============================");
    }
}
```

Class: ArrayOperationsTest

```
/*
 * I affirm that I have carried out the attached academic endeavors with full academic
honesty.
 * Manav Bilakhia (MB)
 */
package assignment;
/**
 * Testing for Array operations that use search and sort
 *
 * @Manav Bilakhia
 */
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class ArrayOperationsTest {

    @Test
    void doubleExists(){
        int[] arr1 = { 9,1,6,3,2,0,1,6 };
        int[] arr2 = { 2,8,6,7,5,2,5,2};
        int[] arr3 = {};
        assertEquals(true,ArrayOperations.doubleExists(arr1));
        assertEquals(false,ArrayOperations.doubleExists(arr2));
        assertEquals(false,ArrayOperations.doubleExists(arr3));
```

```
    }

    @Test
    void moveGivenValue() {
        int[] arr4 = { 2,0,0,2,3,0,0,3 };
        ArrayOperations.moveGivenValue(arr4,0);
        int [] expectedArr4 = {2,2,3,3,0,0,0,0};
        int[] arr5 = { 0 };
        ArrayOperations.moveGivenValue(arr5,0);
        int [] expectedArr5 = {0};
        int[] arr6 = { 1,9,7,7 };
        ArrayOperations.moveGivenValue(arr6,9);
        int [] expectedArr6 = {1,7,7,9};
        assertArrayEquals(expectedArr4,arr4);
        assertArrayEquals(expectedArr5,arr5);
        assertArrayEquals(expectedArr6,arr6);

    }

    @Test
    void sortEvenOdd() {
        int[] arr8 = { 6,6,7,6,5,1,6 };
        ArrayOperations.sortEvenOdd(arr8);
        int[] expectedArr8 = {6, 6, 6, 6, 1, 5, 7};
        int[] arr9 = { 3, 1 };
        ArrayOperations.sortEvenOdd(arr9);
        int[] expectedArr9 = {1,3};
        int[] arr10 = { 8, 6, 2, 4 };
        ArrayOperations.sortEvenOdd(arr10);
        int[] expectedArr10 = {8,6,4,2};
        ArrayOperations.sortEvenOdd(arr10);
        assertArrayEquals(expectedArr8,arr8);
        assertArrayEquals(expectedArr9,arr9);
        assertArrayEquals(expectedArr10,arr10);
    }
}
```

B. Sample output 1

I. Describe your test 1: Checking the double exists method

II. Text output 1:

doubleExists tests

doubleExists(arr1)
9 1 6 3 2 0 1 6
true

doubleExists(arr2)
0 5 3 2 7
false

doubleExists(arr3)
false

III. Screenshot 1:

```
doubleExists tests

doubleExists(arr1)
 9 1 6 3 2 0 1 6
true

doubleExists(arr2)
 0 5 3 2 7
false

doubleExists(arr3)
false
```

C. Sample output 2

I. Describe your test 2: Checking the moveGivenValue method

II. Text output 2:

moveGivenValue tests

moveGivenValue(arr4, 0)
Before: 2 0 0 2 3 0 0 3
After: 2 2 3 3 0 0 0 0

moveGivenValue(arr5, 0)
Before: 0
After: 0

moveGivenValue(arr6, 0)
Before: 1 9 7 7
After: 1 9 7 7

moveGivenValue(arr7, 1)
Before: 1 8 4 5 8 5 1
After: 8 4 5 8 5 1 1

III. Screenshot 2:

```
moveGivenValue tests

moveGivenValue(arr4, 0)
Before: 2 0 0 2 3 0 0 3
After: 2 2 3 3 0 0 0 0

moveGivenValue(arr5, 0)
Before: 0
After: 0

moveGivenValue(arr6, 0)
Before: 1 9 7 7
After: 1 9 7 7

moveGivenValue(arr7, 1)
Before: 1 8 4 5 8 5 1
After: 8 4 5 8 5 1 1
```

D. Sample output 3

I. Describe your test 3: Checking the sortEvenOdd method

II. Text output 3:

sortEvenOdd tests

sortEvenOdd(arr8)
Before: 6 6 7 6 5 1 6
After: 6 6 6 6 1 5 7

sortEvenOdd(arr9)
Before: 3 1
After: 1 3

sortEvenOdd(arr10)
Before: 8 6 2 4
After: 8 6 4 2

III. Screenshot 3:

```
sortEvenOdd tests

sortEvenOdd(arr8)
Before: 6 6 7 6 5 1 6
After: 6 6 6 6 1 5 7

sortEvenOdd(arr9)
Before: 3 1
After: 1 3

sortEvenOdd(arr10)
Before: 8 6 2 4
After: 8 6 4 2
```