FALL Semester 2022-23

ITA5002 - Problem Solving with Data Structures and Algorithms Lab

LAB DIGITAL ASSESSMENT – II

Name: Kamran Ansari

Reg No: 22MCA0223

Q1. Try to create an array of N elements as input which consists of Positive, Negative numbers and also the duplicate values. Perform any sorting procedure to get output as two sorted array one with positive numbers and one with negative numbers and print the number of comparisons made for each sorted array and aslo the say count for the each the duplicate values. Exapmle : input[15]= { 10,4,-3,-1,0,4,3,-15,-8,4,-1,9,3,1,7} Output1[15]={-1,-1,-3,-8,-15} and print Number of Comparisons Output2[15]={0,1,3,3,4,4,4,7,9,10} and print Number of Comparisons -1 has been present twice + 3 has been present twice + 4 has been present thrice

Count.java

```
public class Count {
    public int value;
    public int count;

    public Count(int value) {
        this.value = value;
        this.count = 1;
    }

    @Override
    public String toString() {
```

```
        return this.value + " has been present " + this.count
+ " times";
    }
}
```

CountList.java

```
public class CountList {
    private Count[] list;
    private int size;
    private int length;

    public CountList(int size) {
        this.size = size;
        list = new Count[this.size];
        length = 0;
    }

    public void incrementCount(int value) {
        for (int i = 0; i < length; i++) {
            if (list[i].value == value) {
                list[i].count++;
                return;
            }
        }

        list[length] = new Count(value);
        length++;
    }

    @Override
    public String toString() {
```

```
        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < length; i++) {
            if (list[i].count >= 2) {
                sb.append(list[i].toString() + "\n");
            }
        }

        return sb.toString();
    }
}
```

Main.java

```
import java.util.Scanner;

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

        System.out.print("Enter array size: ");
        int arraySize = Integer.valueOf(scan.nextLine());

        int[] array = new int[arraySize];
        CountList cl = new CountList(arraySize);

        for (int i = 0; i < arraySize; i++) {
            System.out.println("Enter value of " + (i + 1) + "
element: ");
            array[i] = Integer.valueOf(scan.nextLine());
            cl.incrementCount(array[i]);
```

```
        }

        int[] negativeOutputArray = new int[arraySize];
        int negativeOutputArrayLength = 0;

        int[] positiveOutputArray = new int[arraySize];
        int positiveOutputArrayLength = 0;

        for (int i = 0; i < arraySize; i++) {
            if (array[i] < 0) {
negativeOutputArray[negativeOutputArrayLength++] = array[i];
            } else {
positiveOutputArray[positiveOutputArrayLength++] = array[i];
            }
        }

        System.out.println("\nPositive numbers array before
sorting:");
        printArray(positiveOutputArray,
positiveOutputArrayLength);
        System.out.println("\nNegative numbers array before
sorting:");
        printArray(negativeOutputArray,
negativeOutputArrayLength);

        System.out.println("");

        int numberOfComparisons;

        System.out.println("\nPositive numbers array after
sorting:");
```

```
        numberOfComparisons =
insertionSortAsc(positiveOutputArray,
positiveOutputArrayLength);

        printArray(positiveOutputArray,
positiveOutputArrayLength);

        System.out.println("\nNumber of Comparisons made
during sorting : " + numberOfComparisons);


        System.out.println("\nNegative numbers array after
sorting:");

        numberOfComparisons =
insertionSortDesc(negativeOutputArray,
negativeOutputArrayLength);

        printArray(negativeOutputArray,
negativeOutputArrayLength);

        System.out.println("\nNumber of Comparisons made
during sorting : " + numberOfComparisons);


        System.out.println("");

        System.out.println("Duplicate Values:");

        System.out.println(cl);

    }


    public static int insertionSortAsc(int[] arr, int n) {

        int numberOfComparisons = 0;


        for (int i = 1; i < n; i++) {

            int key = arr[i];

            int j = i - 1;


            while (j >= 0 && arr[j] > key) {

                numberOfComparisons++;

                arr[j + 1] = arr[j];

                j = j - 1;
```

```
        }

        arr[j + 1] = key;
    }

    return numberOfComparisons + 1;
}

public static int insertionSortDesc(int[] arr, int n) {
    int numberOfComparisons = 0;

    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

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

        arr[j + 1] = key;
    }

    return numberOfComparisons + 1;
}

public static void printArray(int arr[], int length) {
    for (int i = 0; i < length; i++) {
        System.out.print(arr[i] + " ");
    }
```

```
    }
}
```

Output

```
Enter array size: 15
Enter value of 1 element:
10
Enter value of 2 element:
4
Enter value of 3 element:
-3
Enter value of 4 element:
-1
Enter value of 5 element:
0
Enter value of 6 element:
4
Enter value of 7 element:
3
Enter value of 8 element:
-15
Enter value of 9 element:
-8
Enter value of 10 element:
4
Enter value of 11 element:
-1
Enter value of 12 element:
9
```

```
Enter value of 13 element:
3
Enter value of 14 element:
1
Enter value of 15 element:
7

Positive numbers array before sorting:
10 4 0 4 3 4 9 3 1 7
Negative numbers array before sorting:
-3 -1 -15 -8 -1

Positive numbers array after sorting:
0 1 3 3 4 4 4 7 9 10
Number of Comparisons made during sorting : 24

Negative numbers array after sorting:
-1 -1 -3 -8 -15
Number of Comparisons made during sorting : 6

Duplicate Values:
4 has been present 3 times
-1 has been present 2 times
3 has been present 2 times
```

Q2. Perform Binary Search for the given input

A[14]={11,22,33,44,55,66,77,88,99,111,222,333,444}

And try to print the position of the number present in the upper half starting from 1 to n N is the middle element.

After middle element again the position should start from 1 to n. Here N is the last element.

Example mid = (0+14)/2 ==7 middle element is 88

Numbers 33 present in the third position

88 is present in the ninth position

99 is present in first position

333 is present in fourth position

Main.java

```
import java.util.Scanner;


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


        System.out.println("Enter array size:");
        int arraySize = Integer.valueOf(scan.nextLine());


        int[] array = new int[arraySize];


        System.out.println("Enter array (in ascending order):");
        for (int i = 0; i < arraySize; i++) {
            System.out.println("Enter value of " + (i + 1) + "
element: ");
            array[i] = Integer.valueOf(scan.nextLine());
        }
```

```
        System.out.println("Array: ");
        for (int i = 0; i < arraySize; i++) {
            System.out.print(array[i] + " ");
        }

        System.out.println("");
        System.out.println("Enter value to search for:");
        int value = Integer.valueOf(scan.nextLine());

        int position = binSearch(array, value, 0, arraySize - 1);

        System.out.println("");
        if (position == -1) {
            System.out.println("Element not found");
        } else {
            System.out.println("Element is present at position " +
position);
        }
    }

    public static int binSearch(int[] arr, int value, int min, int
max) {
        int mid = (min + max) / 2;

        if (min > max) {
            return -1;
        }

        System.out.println("");
        System.out.println("Searching for element in sub array:");
        for (int i = min; i <= max; i++) {
            System.out.print(arr[i] + " ");
        }
```

```
        if (arr[mid] == value) {
            return mid - min + 1;
        } else if (arr[mid] > value) {
            return binSearch(arr, value, min, mid);
        } else {
            return binSearch(arr, value, mid + 1, max);
        }
    }
}
```

Output

```
Enter array size:
13
Enter array (in ascending order):
Enter value of 1 element:
11
Enter value of 2 element:
22
Enter value of 3 element:
33
Enter value of 4 element:
44
Enter value of 5 element:
55
Enter value of 6 element:
66
Enter value of 7 element:
77
Enter value of 8 element:
88
Enter value of 9 element:
99
Enter value of 10 element:
111
Enter value of 11 element:
222
```

```
Enter value of 12 element:
333
Enter value of 13 element:
444
Array:
11 22 33 44 55 66 77 88 99 111 222 333 444
Enter value to search for:
77

Searching for element in sub array:
11 22 33 44 55 66 77 88 99 111 222 333 444
Element is present at position 7
```

```
Array:
11 22 33 44 55 66 77 88 99 111 222 333 444
Enter value to search for:
222

Searching for element in sub array:
11 22 33 44 55 66 77 88 99 111 222 333 444
Searching for element in sub array:
88 99 111 222 333 444
Searching for element in sub array:
222 333 444
Searching for element in sub array:
222 333
Element is present at position 1
```

```
Array:
11 22 33 44 55 66 77 88 99 111 222 333 444
Enter value to search for:
111

Searching for element in sub array:
11 22 33 44 55 66 77 88 99 111 222 333 444
Searching for element in sub array:
88 99 111 222 333 444
Element is present at position 3
```

```
Array:
11 22 33 44 55 66 77 88 99 111 222 333 444
Enter value to search for:
44

Searching for element in sub array:
11 22 33 44 55 66 77 88 99 111 222 333 444
Searching for element in sub array:
11 22 33 44 55 66 77
Element is present at position 4
```