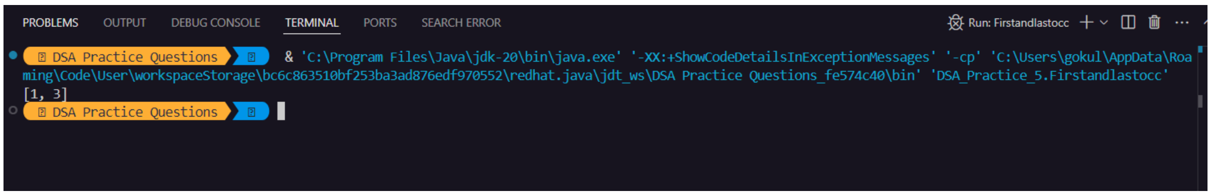
1.Given a sorted array **arr** with possibly some duplicates, the task is to find the first and last occurrences of an element **x** in the given array.
Note: If the number **x** is not found in the array then return both the indices as -1.

Solution

```
package DSA_Practice_5;

import java.util.ArrayList;

public class Firstandlastocc {
  public static void main(String[] args) {
    int[] arr = {1, 2, 2, 2, 3, 4, 5};
    int x = 2;
    ArrayList<Integer> res = find(arr,x);
    System.out.println(res.toString());
  }
   public static ArrayList<Integer> find(int arr[], int x) {
    ArrayList<Integer> res = new ArrayList<>();
    res.add(firstocc(arr,x));
    res.add(lastocc(arr,x));
    return res;
   }
   public static int firstocc(int[]arr,int x){
     int left =0, right = arr.length-1,result=-1;
     while(left<=right){
       int mid = (left+right)/2;
       if(arr[mid]==x){
         result = mid;
         right = mid-1;
       }
       else if(arr[mid]<x) left=mid+1;
       else right=mid-1;
```

```
        }
        return result;
    }
    public static int lastocc(int[]arr,int x){
        int left =0, right = arr.length-1,result=-1;
        while(left<=right){
            int mid = (left+right)/2;
            if(arr[mid]==x){
                result = mid;
                left = mid+1;
            }
            else if(arr[mid]<x) left=mid+1;
            else right=mid-1;
        }
        return result;
    }
}
```

Output:



```
● DSA Practice Questions  & 'C:\Program Files\Java\jdk-20\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\gokul\AppData\Roa
ming\Code\User\workspaceStorage\bc6c863510bf253ba3ad876edf970552\redhat.java\jdt_ws\DSA Practice Questions_fe574c40\bin' 'DSA_Practice_5.Firstandlastocc'
[1, 3]
○ DSA Practice Questions
```

Time Complexit:O(log n)

Space Complexity: O(1)

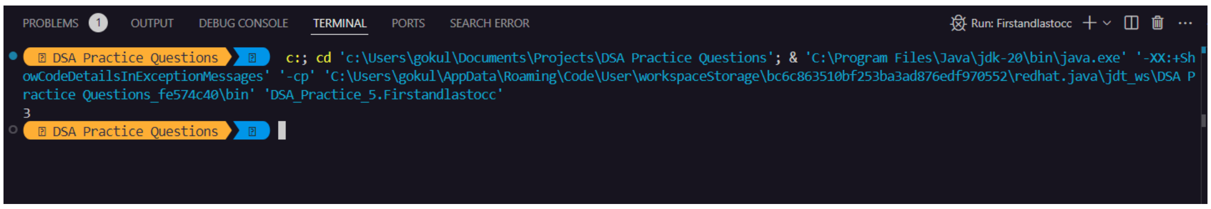
2. Given a **sorted array, arr[]** containing only **0s** and **1s**, find the **transition point**, i.e., the **first index** where **1** was observed, and **before that**, only 0 was observed. If **arr** does not have any **1**, return **-1**. If array does not have any **0**, return **0**.

Solution:

```
package DSA_Practice_5;


import java.util.ArrayList;
```

```
public class Firstandlastocc {
    public static void main(String[] args) {
        int[] arr = {0, 0, 0, 1, 1};
        int res = transitionPoint(arr);
        System.out.println(res);
    }
   public static int transitionPoint(int arr[]) {
        int left=0,right=arr.length-1,res =-1;
        while(left<=right){
            int mid = (left+right)/2;
            if(arr[mid]==1){
                res = mid;
                right = mid-1;
            }
            else if(arr[mid]<1) left = mid+1;
            else right = mid-1;
        }
        return res;
    }
 }
```

Output:



```
DSA Practice Questions   c:; cd 'c:\Users\gokul\Documents\Projects\DSA Practice Questions'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\gokul\AppData\Roaming\Code\User\workspaceStorage\bc6c863510bf253ba3ad876edf970552\redhat.java\jdt_ws\DSA Practice Questions_fe574c40\bin' 'DSA_Practice_5.Firstandlastocc'
3
DSA Practice Questions
```

Time Complexit:O(log n)

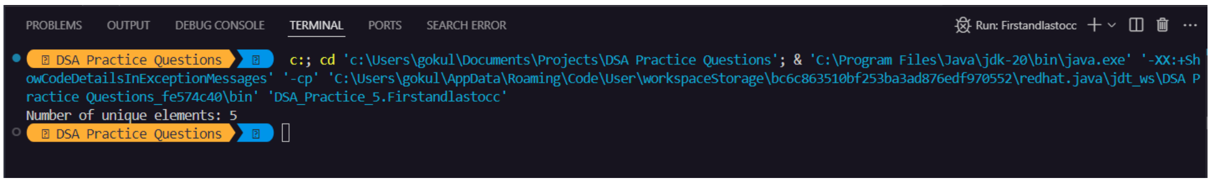
Space Complexity: O(1)

3. Given a **sorted** array **arr.** Return the size of the modified array which contains only distinct elements.
Note:
1. Don't use set or HashMap to solve the problem.
2. You **must** return the modified array **size only** where distinct elements are present and **modify** the original array such that all the distinct elements come at the beginning of the original array.

Solution:

```
package DSA_Practice_5;


import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;


public class Firstandlastocc {

   public static void main(String[] args) {

       List<Integer> arr = new ArrayList<>(Arrays.asList(1, 1, 2, 2, 3, 4, 4, 5));

       int newLength = remove_duplicate(arr);

       System.out.println("Number of unique elements: " + newLength);

   }

  public static int  remove_duplicate(List<Integer> arr) {

       int j=1;

       for(int i=1;i<arr.size();i++){

           if(!arr.get(i).equals(arr.get(j - 1))){ //this comparison !arr.get(i).equals(arr.get(j - 1)) is speciall
for the comapring Integer Values

               arr.set(j,arr.get(i));

               j++;

           }

       }

       return j;

   }

}
```

Output:



```
 DSA Practice Questions   c:; cd 'c:\Users\gokul\Documents\Projects\DSA Practice Questions'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '-XX:+Sh
owCodeDetailsInExceptionMessages' '-cp' 'C:\Users\gokul\AppData\Roaming\Code\User\workspaceStorage\bc6c863510bf253ba3ad876edf970552\redhat.java\jdt_ws\DSA P
ractice Questions_fe574c40\bin' 'DSA_Practice_5.Firstandlastocc'
Number of unique elements: 5
 DSA Practice Questions 
```

Time Complexit:O(n)

Space Complexity: O(1)

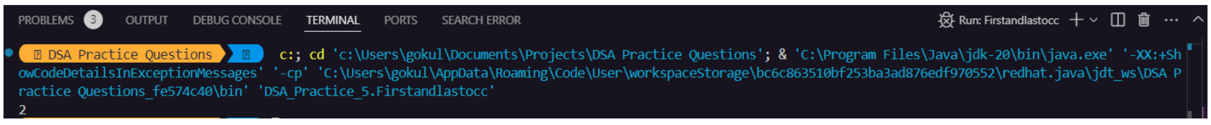
4. Given an array **arr[],** find the first repeating element. The element should occur more than once and the index of its first occurrence should be the smallest.

Note:- The position you return should be according to 1-based indexing.

Solution:

```
package DSA_Practice_5;


import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

import java.util.Map;


public class Firstandlastocc {

    public static void main(String[] args) {

        System.out.println(firstRepeated(new int[] {1, 5, 3, 4, 3, 5, 6}));

    }

   public static int firstRepeated(int[] arr) {

        Map<Integer,int[]> map = new HashMap<>();

        for(int i=0;i<arr.length;i++){

            if(!map.containsKey(arr[i])){

                map.put(arr[i],new int[]{i+1,1});// i-firstindex , 1-for frequency

            }

            else{

                int[] a = map.get(arr[i]);

                a[1]++;
            }
        }
        int res = Integer.MAX_VALUE;
        for(int[] r : map.values()){
            if(r[1]>=2) res = Math.min(res,r[0]);
        }
        return res==Integer.MAX_VALUE?-1:res;
    }
}
```

Output:



```
DSA Practice Questions  c:; cd 'c:\Users\gokul\Documents\Projects\DSA Practice Questions'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '-XX:+Sh
owCodeDetailsInExceptionMessages' '-cp' 'C:\Users\gokul\AppData\Roaming\Code\User\workspaceStorage\bc6c863510bf253ba3ad876edf970552\redhat.java\jdt_ws\DSA P
ractice Questions_fe574c40\bin' 'DSA_Practice_5.Firstandlastocc'
2
```

Time Complexit:O(n)

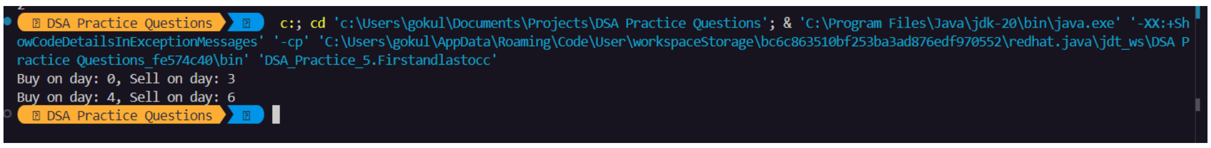
Space Complexity: O(n)

5. The cost of stock on each day is given in an array **A**[] of size **N**. Find all the segments of days on which you buy and sell the stock such that the sum of difference between sell and buy prices is maximized. Each segment consists of indexes of two elements, first is index of day on which you buy stock and second is index of day on which you sell stock.
Note: Since there can be multiple solutions, the driver code will print 1 if your answer is correct, otherwise, it will return 0. In case there's no profit the driver code will print the string "**No Profit**" for a correct solution.

Solution:

```
package DSA_Practice_5;


import java.util.ArrayList;


public class Firstandlastocc {
    public static void main(String[] args) {
        int[] price = {100, 180, 260, 310, 40, 535, 695};
        ArrayList<ArrayList<Integer>> result = stockBuySell(price, price.length);
        for (ArrayList<Integer> interval : result) {
            System.out.println("Buy on day: " + interval.get(0) + ", Sell on day: " + interval.get(1));
        }
    }
    public static ArrayList<ArrayList<Integer> > stockBuySell(int price[], int n) {
        ArrayList<ArrayList<Integer>> list = new ArrayList<>();
        int startindex=0;
        for(int i=1;i<n;i++){
            if(price[i]<price[i-1]){
                if(i-1>startindex){
                ArrayList<Integer> res = new ArrayList<>();
                res.add(startindex);
                res.add(i-1);
                list.add(res);
                }
                startindex =i;
            }
        }
        if(n-1>startindex){
            ArrayList<Integer> res = new ArrayList<>();
                res.add(startindex);
                res.add(n-1);
                list.add(res);
        }
        return list;
    }
  }
```

Output:



```
 DSA Practice Questions   c:; cd 'c:\Users\gokul\Documents\Projects\DSA Practice Questions'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '-XX:+Sh
owCodeDetailsInExceptionMessages' '-cp' 'C:\Users\gokul\AppData\Roaming\Code\User\workspaceStorage\bc6c863510bf253ba3ad876edf970552\redhat.java\jdt_ws\DSA P
ractice Questions_fe574c40\bin' 'DSA_Practice_5.Firstandlastocc'
Buy on day: 0, Sell on day: 3
Buy on day: 4, Sell on day: 6
 DSA Practice Questions  
```

Time Complexit:O(n)

Space Complexity: O(n)