Practice – 5

Pradeep S

CSBS

1)Wave Array

```
import java.util.*;


class WaveArray {
   public static void convertToWave(int[] arr) {
      int i = 0;
      while (i < arr.length - 1) {
         int temp = arr[i];
         arr[i] = arr[i + 1];
         arr[i + 1] = temp;
         i += 2;
      }
   }



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

      System.out.println("Enter the number of elements in the array:");
      int n = sc.nextInt();
           System.out.println("Enter the elements of the array");
```

```
    int[] arr = new int[n];

    for (int i = 0; i < n; i++) {
        arr[i] = sc.nextInt();
    }
        convertToWave(arr);

    for (int i : arr) {
        System.out.print(i + " ");
    }
  }
}
```

Input: n=5 **,** arr[] = [1, 2, 3, 4, 5]

Output: [2, 1, 4, 3, 5]

Explanation: Array elements after sorting it in the waveform are 2, 1, 4, 3, 5.

2) Remove Duplicates From An Sorted Array

```
import java.util.*;
class Solution {

  public static int remove_duplicate(List<Integer> arr) {
    if (arr.size() == 0) return 0;

    int i = 0;
    for (int j = 1; j < arr.size(); j++) {
      if (!arr.get(j).equals(arr.get(i))) {
        i++;
```

```
            arr.set(i, arr.get(j));
          }
        }
        return i + 1;
    }
        public static void main(String[] args){



        Scanner sc = new Scanner(System.in);


        System.out.println("Enter the number of elements in the array:");
        int n = sc.nextInt();


        List<Integer> arr = new ArrayList<>();
        System.out.println("Enter the sorted elements of the array:");
        for (int i = 0; i < n; i++) {
          arr.add(sc.nextInt());
        }
        System.out.println(remove_duplicate(arr));
    }


}
```

Input: arr = [2, 2, 2, 2, 2]

Output: [2]

Explanation: After removing all the duplicates only one instance of 2 will remain i.e. [2] so modified array will contains 2 at first position and you should return 1 after modifying the array, the driver code will print the modified array elements.

3)Find Transition Point

```
import java.util.*;

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

    System.out.println("Enter the number of elements in the array:");
    int n = sc.nextInt();

    int[] arr = new int[n];
    System.out.println("Enter the elements of the array (sorted 0s and 1s):");
    for (int i = 0; i < n; i++) {
      arr[i] = sc.nextInt();
    }

    System.out.println(transitionPoint(arr));
  }

  static int transitionPoint(int arr[]) {
    for (int i = 0; i < arr.length; i++) {
      if (arr[i] == 1) {
        return i;
      }
    }
```

```
        return -1;
    }
}
```

Input: arr[] = [0, 0, 0, 1, 1]

Output: 3

Explanation: index 3 is the transition point where 1 begins.

4)First and Last Occurrence

```
import java.util.*;

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

        System.out.println("Enter the number of elements in the array:");
        int n = sc.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.println("Enter the target element to find:");
        int x = sc.nextInt();
```

```
    ArrayList<Integer> result = find(arr, x);

    System.out.println("First and last positions of target element:");
    System.out.println(result.get(0) + " " + result.get(1));
}

static ArrayList<Integer> find(int arr[], int x) {
    int X = Integer.MAX_VALUE, Y = -1;
    boolean isValid = false;
    int n = arr.length;

    for (int i = 0; i < n; i++) {
        int j = n - i - 1;
        if (arr[i] == x) {
            X = Math.min(X, i);
            isValid = true;
            Y = Math.max(Y, j);
        }
    }

    if (!isValid) {
        X = -1;
    }

    ArrayList<Integer> al = new ArrayList<>();
    al.add(X);
    al.add(Y);
```

```
        return al;
    }
}
```

Input: arr[] = [1, 3, 5, 5, 5, 5, 67, 123, 125], x = 5

Output: [2, 5]

Explanation: First occurrence of 5 is at index 2 and last occurrence of 5 is at index 5

5)First Repeating Element

```
import java.util.*;

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

        System.out.println("Enter the number of elements in the array:");
        int n = sc.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        int result = firstRepeated(arr);
```

```
        if (result == -1) {
            System.out.println("No repeating elements found.");
        } else {
            System.out.println("Position of the first repeating element: " + result);
        }
    }



    public static int firstRepeated(int[] arr) {
        int v = -1;
        Set<Integer> a = new HashSet<>();

        for (int i = arr.length - 1; i >= 0; i--) {
            if (a.contains(arr[i])) {
                v = i;
            } else {
                a.add(arr[i]);
            }
        }

        if (v != -1) {
            return v + 1;
        }
        return -1;
    }
}
```

Input: arr[] = [1, 5, 3, 4, 3, 5, 6]

Output: 2

Explanation: 5 appears twice and its first appearance is at index 2 which is less than 3 whose first the occurring index is 3.

6)Buy and Sell Stock

```
import java.util*;


class MaxProfit1 {
  static int maxProfit(int[] prices) {
    int n = prices.length;
    int minsofar = prices[0];
    int result = 0;
    for (int i = 0; i < n; i++) {
      minsofar = Math.min(minsofar, prices[i]);
      result = Math.max(result, prices[i] - minsofar);
    }
    return result;
  }

  public static void main(String[] args) {
        Scanner sc = new Scanner(System.in)
        Int n = sc.nextInt();
        Int[] arr = new int[n];
        for(int i=0;i<n;i++){
                arr[i] = sc.nextInt();
```

```
}

    System.out.println(maxProfit(prices));
  }
}
```

Input:

n = 7

arr[] = {100,180,260,310,40,535,695}

Output:

1

Explanation:

One possible solution is (0 3) (4 6)

We can buy stock on day 0,

and sell it on 3rd day, which will

give us maximum profit. Now, we buy

stock on day 4 and sell it on day 6.

7)Coin Change

```
Import java.util*;
class CoinChange {
  static long count(int coins[], int n, int sum) {
    int dp[] = new int[sum + 1];
    dp[0] = 1;
    for (int i = 0; i < n; i++)
      for (int j = coins[i]; j <= sum; j++)
        dp[j] += dp[j - coins[i]];
    return dp[sum];
```

```
  }

  public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        Int n = sc.nextInt();
        Int[] coins = new int[n];
        for(int i=0;i<n;i++){
              arr[i] = sc.nextInt();
        }
    i
    int sum = sc.nextInt();
    System.out.println(count(coins, n, sum));
  }
}
```

Input: *sum = 4, coins[] = {1,2,3}*
Output: *4*
Explanation: *there are four solutions: {1, 1, 1, 1}, {1, 1, 2}, {2, 2} and {1, 3}*

8)Minimum Index

```
Import java.util*;
public class FindMaximum {
  int max(int x, int y) {
    return x > y ? x : y;
  }

  int min(int x, int y) {
    return x < y ? x : y;
  }
```

```
int maxIndexDiff(int arr[], int n) {
   int maxDiff;
   int i, j;
   int RMax[] = new int[n];
   int LMin[] = new int[n];

   LMin[0] = arr[0];
   for (i = 1; i < n; ++i)
      LMin[i] = min(arr[i], LMin[i - 1]);

   RMax[n - 1] = arr[n - 1];
   for (j = n - 2; j >= 0; --j)
      RMax[j] = max(arr[j], RMax[j + 1]);

   i = 0;
   j = 0;
   maxDiff = -1;
   while (j < n && i < n) {
      if (LMin[i] <= RMax[j]) {
         maxDiff = max(maxDiff, j - i);
         j = j + 1;
      } else
         i = i + 1;
   }
   return maxDiff;
}
```

```
    public static void main(String[] args) {
        FindMaximum max = new FindMaximum();
        Scanner sc = new Scanner(System.in);
        Int n = sc.nextInt();
        Int[] arr = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = sc.nextInt();
        }
      int maxDiff = max.maxIndexDiff(arr, n);
      System.out.println(maxDiff);
    }
}
```

Input: n = 9,arr[] = [34, 8, 10, 3, 2, 80, 30, 33, 1]

Output: 6

Explanation: In the given array arr[1] < arr[7] satisfying the required condition(arr[i] ≤ arr[j]) thus giving the maximum difference of j - i which is 6(7-1).