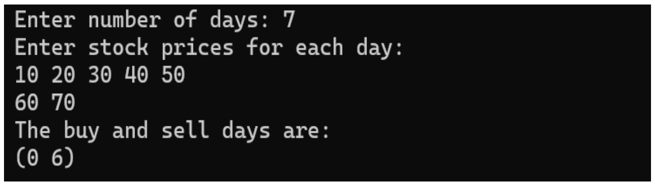
PRACTICE SET – 4

DSA

1.STOCK BUY AND SELL

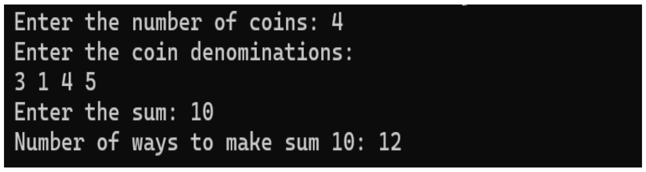
```
import java.util.ArrayList;
import java.util.Scanner;

public class Buy {
    ArrayList<ArrayList<Integer>> stockBuySell(int A[], int n) {
        ArrayList<ArrayList<Integer>> result = new ArrayList<>();
        int i = 0;
        while (i < n - 1) {
            while (i < n - 1 && A[i + 1] <= A[i]) {
                i++;
            }
            if (i == n - 1) {
                break;
            }
            int buy = i++;
            while (i < n && A[i] >= A[i - 1]) {
                i++;
            }
            int sell = i - 1;
            ArrayList<Integer> pair = new ArrayList<>();
            pair.add(buy);
            pair.add(sell);
            result.add(pair);
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of days: ");
        int n = sc.nextInt();
        int[] A = new int[n];
        System.out.println("Enter stock prices for each day: ");
        for (int i = 0; i < n; i++) {
            A[i] = sc.nextInt();
        }
        Buy solution = new Buy();
        ArrayList<ArrayList<Integer>> result = solution.stockBuySell(A, n);
        if (result.isEmpty()) {
            System.out.println("No Profit");
        } else {

            System.out.println("The buy and sell days are: ");
            for (ArrayList<Integer> pair : result) {
                System.out.println("(" + pair.get(0) + " " + pair.get(1) + ")");
            }
        }
        sc.close();
    }
}
```



```
Enter number of days: 7
Enter stock prices for each day:
10 20 30 40 50
60 70
The buy and sell days are:
(0 6)
```

2.COIN CHANGE(COUNT WAYS)

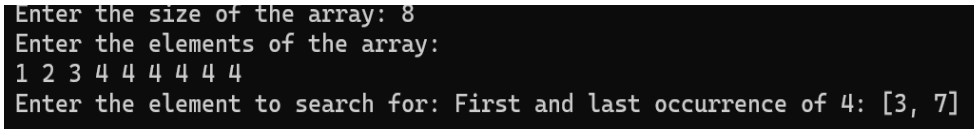
```
import java.util.Scanner;

public class Coin {

    public static int countWays(int[] coins, int sum) {
        int[] dp = new int[sum + 1];
        dp[0] = 1;

        for (int coin : coins) {
            for (int i = coin; i <= sum; i++) {
                dp[i] += dp[i - coin];
            }
        }

        return dp[sum];
    }

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

        System.out.print("Enter the number of coins: ");
        int n = sc.nextInt();

        int[] coins = new int[n];
        System.out.println("Enter the coin denominations: ");
        for (int i = 0; i < n; i++) {
            coins[i] = sc.nextInt();
        }

        System.out.print("Enter the sum: ");
        int sum = sc.nextInt();


        int result = countWays(coins, sum);

        System.out.println("Number of ways to make sum " + sum + ": " + result);

        sc.close();
    }
}
```



```
Enter the number of coins: 4
Enter the coin denominations:
3 1 4 5
Enter the sum: 10
Number of ways to make sum 10: 12
```

3.FIRST AND LAST OCCURENCES

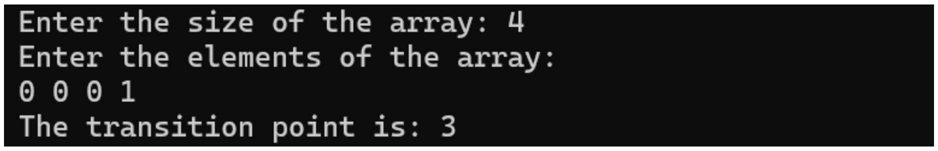
```
import java.util.Scanner;

public class FindOccurrences {

    public static int[] findFirstAndLast(int[] arr, int x) {
        int[] result = new int[2];
        result[0] = -1;
        result[1] = -1;

        int low = 0, high = arr.length - 1;

        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == x) {
                result[0] = mid;
                result[1] = mid;
                while (result[0] > 0 && arr[result[0] - 1] == x) {
                    result[0]--;
                }
                while (result[1] < arr.length - 1 && arr[result[1] + 1] == x) {
                    result[1]++;
                }
                break;
            } else if (arr[mid] < x) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }

        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of 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.print("Enter the element to search for: ");
        int x = sc.nextInt();
        int[] result = findFirstAndLast(arr, x);
        System.out.println("First and last occurrence of " + x + ": [" + result[0] + ", " + result[1] + "]");
        sc.close();
    }
}
```



```
Enter the size of the array: 8
Enter the elements of the array:
1 2 3 4 4 4 4 4 4
Enter the element to search for: First and last occurrence of 4: [3, 7]
```

4.FIND TRANSITION POINT

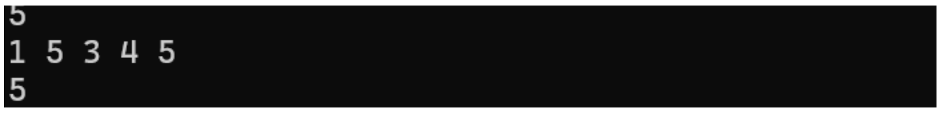
```
import java.util.Scanner;

public class Point {

    public static int findTransitionPoint(int[] arr) {
        int low = 0, high = arr.length - 1;

        if (arr[low] == 1) {
            return 0;
        }
        if (arr[high] == 0) {
            return -1;
        }

        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == 1 && (mid == 0 || arr[mid - 1] == 0)) {
                return mid;
            } else if (arr[mid] == 0) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }

        return -1;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of 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 = findTransitionPoint(arr);
        System.out.println("The transition point is: " + result);
        sc.close();
    }
}
```



```
Enter the size of the array: 4
Enter the elements of the array:
0 0 0 1
The transition point is: 3
```

5.FIRST REPEATING ELEMENT

```
import java.util.*;

class Repeat {
   int firstRepeatingElement(int arr[]) {
      Set<Integer> seen = new HashSet<>();
      for (int i = 0; i < arr.length; i++) {
         if (seen.contains(arr[i])) {
            return i + 1;
         }
         seen.add(arr[i]);
      }
      return -1;
   }

   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();
      }

      Repeat solution = new Repeat();
      System.out.println(solution.firstRepeatingElement(arr));

      sc.close();
   }
}
```

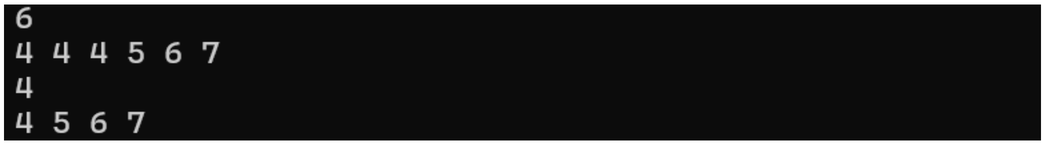


```
5
1 5 3 4 5
5
```

6.REMOVE DUPLICATES FROM SORTED ARRAY

```
import java.util.*;

class Remove {
   int removeDuplicates(int arr[]) {
      if (arr.length == 0) {
         return 0;
```

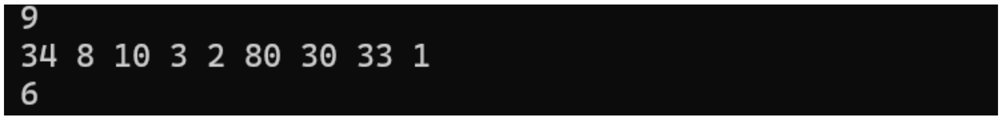
```
        }

        int uniqueIndex = 1;
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] != arr[i - 1]) {
                arr[uniqueIndex] = arr[i];
                uniqueIndex++;
            }
        }
        return uniqueIndex;
    }

    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();
        }

       Remove solution = new Remove();
        int newSize = solution.removeDuplicates(arr);
        System.out.println(newSize);

        for (int i = 0; i < newSize; i++) {
            System.out.print(arr[i] + " ");
        }

        sc.close();
    }
}
```



```
6
4 4 4 5 6 7
4
4 5 6 7
```

6.MAXIMUM INDEX

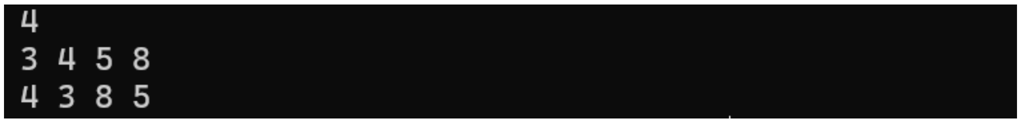
```
import java.util.*;

class Max {
    int maxIndexDiff(int arr[]) {
        int n = arr.length;
        int[] leftMin = new int[n];
        int[] rightMax = new int[n];

        leftMin[0] = arr[0];
        for (int i = 1; i < n; i++) {
            leftMin[i] = Math.min(arr[i], leftMin[i - 1]);
        }

        rightMax[n - 1] = arr[n - 1];
        for (int i = n - 2; i >= 0; i--) {
            rightMax[i] = Math.max(arr[i], rightMax[i + 1]);
        }

        int i = 0, j = 0, maxDiff = -1;
        while (i < n && j < n) {
            if (leftMin[i] < rightMax[j]) {
                maxDiff = Math.max(maxDiff, j - i);
                j++;
            } else {
                i++;
            }
        }
        return maxDiff;
    }

    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();
        }

        Max solution = new Max();
        System.out.println(solution.maxIndexDiff(arr));

        sc.close();
    }
}
```



```
9
34 8 10 3 2 80 30 33 1
6
```

10.WAVE ARRAY

```
import java.util.Scanner;

class Wave {
    void waveArray(int arr[]) {
        int n = arr.length;

        // Traverse the array in steps of 2, and swap adjacent elements
        for (int i = 0; i < n - 1; i += 2) {
            // Swap arr[i] and arr[i+1] to satisfy the condition for wave-like array
            if (arr[i] < arr[i + 1]) {
                int temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
            }
            // If i > 0 and arr[i-1] is smaller than arr[i], swap them again
            if (i > 0 && arr[i] < arr[i - 1]) {
                int temp = arr[i];
                arr[i] = arr[i - 1];
                arr[i - 1] = temp;
            }
        }
    }

    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();
        }

       Wave solution = new Wave();
        solution.waveArray(arr);

        // Print the modified array
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }

        sc.close();
    }
}
```



```
4
3 4 5 8
4 3 8 5
```