Q1: Merge two arrays by satisfying given constraints Given two sorted arrays X[] and Y[] of size m and n each where m >= n and X[] has exactly n vacant cells, merge elements of Y[] in their correct position in array X[], i.e., merge (X, Y) by keeping the sorted order.

```
package techtest;
import java.util.Arrays;

class Main
{

    private static void merge(int[] X, int[] Y, int m, int n)
    {

        int k = m + n + 1;


        while (m >= 0 && n >= 0)
        {

            if (X[m] > Y[n]) {
                X[k--] = X[m--];
            }
            else {
                X[k--] = Y[n--];
            }
        }


        while (n >= 0) {
            X[k--] = Y[n--];
        }


        Arrays.fill(Y, 0);
    }


    public static void rearrange(int[] X, int[] Y)
    {

        if (X.length == 0) {
            return;
        }


        int k = 0;
        for (int value: X)
        {
```

```
            if (value != 0) {
                X[k++] = value;
            }
        }


        merge(X, Y, k - 1, Y.length - 1);
    }

    public static void main (String[] args)
    {

        int[] X = { 0, 2, 0, 3, 0, 5, 6, 0, 0};
        int[] Y = { 1, 8, 9, 10, 15 };


        rearrange(X, Y);


        System.out.println(Arrays.toString(X));
    }
}
```

// output - [1, 2, 3, 5, 6, 8, 9, 10, 15]

Q4:Write a Java Program to find the duplicate characters in a string.

```
package techtest;
import java.util.*;
public class countelement {

       public static void
                countDuplicateCharacters(String str)
                {


                    Map<Character, Integer> map
                        = new HashMap<Character, Integer>();


                    char[] charArray = str.toCharArray();


                    for (char c : charArray) {

                        if (map.containsKey(c)) {
```

```
                map.put(c, map.get(c) + 1);
            }
            else {


                map.put(c, 1);
            }
        }


        for (Map.Entry<Character, Integer> entry :
             map.entrySet()) {

            if (entry.getValue() > 1) {
                System.out.println(entry.getKey()
                                   + " : "
                                   + entry.getValue());
            }
        }
    }


    public static void
    main(String args[])
    {

        String str = "windowsnotworking";


        countDuplicateCharacters(str);
    }


}
```

//output - w : 3
i : 2
n : 3
o : 3

Q2:Find maximum sum path involving elements of given arrays

Given two sorted arrays of integers, find a maximum sum path involving elements of both arrays whose sum is maximum.

We can start from either array, but we can switch between arrays only through its common elements.

```
import java.util.*;

public class Main {

    static int max(int x, int y) {
        return (x > y) ? x : y;
    }

    static int maxPathSum(int[] ar1, int[] ar2, int m, int n) {
        int i = 0, j = 0;

        int result = 0, sum1 = 0, sum2 = 0;

        while (i < m && j < n) {
            if (ar1[i] < ar2[j])
                sum1 += ar1[i++];
            else if (ar1[i] > ar2[j])
                sum2 += ar2[j++];
            else {
                result += max(sum1, sum2) + ar1[i];

                sum1 = 0;
                sum2 = 0;

                i++;
                j++;
            }
        }

        while (i < m)
            sum1 += ar1[i++];
```

```
        while (j < n)
            sum2 += ar2[j++];

        result += max(sum1, sum2);

        return result;
    }

    public static void main(String[] args) {
        int[] ar1 = {3, 6, 7, 8, 10, 12, 15, 18, 100};
        int[] ar2 = {1, 2, 3, 5, 7, 9, 10, 11, 15, 16, 18, 25, 50};
        int m = ar1.length;
        int n = ar2.length;

        // Function call
        System.out.println("Maximum sum path is " + maxPathSum(ar1, ar2, m, n));
    }
}
```

Q3:Write a Java Program to count the number of words in a string using HashMap.

```
 import java.io.*;

import java.util.HashMap;
import java.util.Map;

class GFG {
    public static void main(String[] args)
    {

        String str = "Alice is girl and Bob is boy";

        Map<String, Integer> hashMap = new HashMap<>();

         String[] words = str.split(" ");
```

```
        for (String word : words) {


            Integer integer = hashMap.get(word);

            if (integer == null)

.
                hashMap.put(word, 1);

            else {

                hashMap.put(word, integer + 1);
            }
        }
        System.out.println(hashMap);
    }
}
// output
//{Bob=1, Alice=1, and=1, is=2, girl=1, boy=1}
```