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.

For example,

Input: X[] = { 0, 2, 0, 3, 0, 5, 6, 0, 0 } Y[] = {1, 8, 9, 10, 15 } The vacant cells in X[] is represented by 0

Code:

```
import java.util.Arrays;

public class MergeArrays {
  public static void mergeArrays(int[] X, int[] Y) {
    int m = X.length;
    int n = Y.length;



    int idx = m - 1;
    for (int i = m - 1; i >= 0; i--) {
      if (X[i] != 0) {
        X[idx--] = X[i];
      }
    }



    int i = 0;
    int j = n;

    int k = 0;


    while (i < n && j < m) {

        if (Y[i] < X[j]) {

            X[k++] = Y[i++];

        } else {

            X[k++] = X[j++];

        }

    }



    while (i < n) {

        X[k++] = Y[i++];

    }

}


public static void main(String[] args) {

    int[] X = {0, 2, 0, 3, 0, 5, 6, 0, 0};

    int[] Y = {1, 8, 9, 10, 15};


    mergeArrays(X, Y);


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

}
```

```
}
```

Output:

```
[1, 2, 3, 5, 6, 8, 9, 10, 15]
```

2: 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.

For example,

Input: X = { 3, 6, 7, 8, 10, 12, 15, 18, 100 } Y = {1, 2, 3, 5, 7, 9, 10, 11, 15, 16, 18, 25, 50}

The maximum sum path is: 1 -> 2 >3->6-7-9->10-> 12 > 15->16 -> 18 -> 100

The maximum sum is 199

********CODE*************

```
public class MaximumSumPath {
    public static void main(String[] args) {
        int[] X = {3, 6, 7, 8, 10, 12, 15, 18, 100};
        int[] Y = {1, 2, 3, 5, 7, 9, 10, 11, 15, 16, 18, 25, 50};

        int sumX = 0, sumY = 0, maxSum = 0;
        int indxX = 0, indxY = 0;

        while (indxX < X.length && indxY < Y.length) {
            if (X[indxX] < Y[indxY]) {
                sumX =  sumX+ X[indxX++];
            }

            else if (X[indxX] > Y[indxY]) {

                sumY += Y[indxY++];

            }

            else {

                // When elements are equal, take the maximum sum till now and add current common
element

                maxSum = maxSum+ Math.max(sumX, sumY) + X[indxX];

                sumX = 0;

                sumY = 0;

                indxX++;

                indxY++;

            }

        }

 // Add remaining elements of X and Y

        while (indxX < X.length) {

            sumX = sumX + X[indxX++];

 }

while (indxY < Y.length) {

            sumY = sumY+ Y[indxY++];

 }

 // Add the maximum sum of remaining elements

        maxSum = maxSum + Math.max(sumX, sumY);

  System.out.println("Maximum sum : " + maxSum);

    }

}//OUTPUT //            Maximum sum : 199
```

3:Q3:Write a Java Program to count themumber of words in a string using HashMap

```
import java.util.HashMap;

public class WordCount {
    public static void main(String[] args) {
        String inputString = "Hello Everyone ! Hello guys Good morning everyone";
        String[] words = inputString.trim().split("\\s+");

        // Create a HashMap to store word counts
        HashMap<String, Integer> wordCountMap = new HashMap<>();
         for (String word : words) {
            String lowercaseWord = word.toLowerCase();

            wordCountMap.put(lowercaseWord, wordCountMap.getOrDefault(lowercaseWord, 0) + 1);
        }
         System.out.println("Word count in the string:");
        for (String word : wordCountMap.keySet()) {
            System.out.println(word + ": " + wordCountMap.get(word));
        }
    }
}
```

```
/** OUTPUT ***/
Word count in the string:
```

```
!: 1

everyone: 2

guys: 1

hello: 2

good: 1

morning: 1

string.: 1

world!: 1

sample: 1

hello: 1
```

Q4:Write a java program to find the duplicate characters in a string

```
public class DuplicateCharacters {

    public static void main(String[] args) {

      String string1 = "abcdabcaba";

      int count;

      char string[] = string1.toCharArray();


      System.out.println("Duplicate characters in a given string: ");


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

         count = 1;
```

```
          for(int j = i+1; j <string.length; j++) {
             if(string[i] == string[j] && string[i] != ' ') {
                count++;


                string[j] = '0';
             }
          }


          if(count > 1 && string[i] != '0' )
             System.out.println(string[i]);
}
}
}
```

```
**********OUTPUT**********
Duplicate characters in a given string:
a
b
c
```