```java
//Tyler Kim
//ACSL Contest 2 2019 - 2020
//Intermediate Division
//ACSL Sameness Factor


import java.io.FileNotFoundException;
import java.lang.reflect.Array;
import java.util.Scanner;
import java.util.List;
import java.util.ArrayList;
import java.io.File;
import java.io.FileNotFoundException;

public class TKim_2_ACSLSamenessFactor_201920 {

    public static void main(String args[]) throws FileNotFoundException {
        //Initialization
        Scanner input = new Scanner(new
        File("C:\\Users\\tyler\\dev\\TylerKimJavaHomework\\Competition\\src\\TKim_2_ACSLS
        amenessFactor_201920_SampleInputs"));

        while (input.hasNext()) {
            String[] tempArr = input.nextLine().split(" ");
            String text1 = tempArr[0];
            String text2 = tempArr[1];

            System.out.println(text1 + " " + text2);

            //run sameness calculation
            int output = samenessCalculator(text1, text2);
            System.out.println(output);

        }
    }

    public static int samenessCalculator(String text1, String text2) {
        boolean hasAccomplishedTask = false;
        String finalText1 = text1;
        String finalText2 = text2;
        int finalValue = 0;

        while(true) {
            ArrayList<String> purge = positionAndIdentityTest(finalText1, finalText2);
            purge = overAndIdentityTest(purge.get(0), purge.get(1));

            if(purge.get(0).equals(finalText1) && purge.get(1).equals(finalText2)) {
                break;
            }

            finalText1 = purge.get(0);
            finalText2 = purge.get(1);

        }

        finalValue = ASFCalculations(finalText1, finalText2);

        return finalValue;
    }

    //same position and type
    public static ArrayList<String> positionAndIdentityTest(String text1, String text2) {
        StringBuilder stringBuilder1 = new StringBuilder();
        StringBuilder stringBuilder2 = new StringBuilder();
        int currentIndex = 0;

        ArrayList<String> finalArrayList = new ArrayList<>();

        ArrayList<String> firstText = convertStringToArray(text1);
```

```
68          ArrayList<String> secondText = convertStringToArray(text2);
69
70          for (int i = 0; i < firstText.size(); i++) {
71              //tester
72              try {
73
74                  if (!firstText.get(i).equals(secondText.get(i))) {
75                      stringBuilder1.append(firstText.get(i));
76                      stringBuilder2.append(secondText.get(i));
77                  }
78
79              } catch (IndexOutOfBoundsException e) {
80                  break;
81              }
82
83              currentIndex = i;
84          }
85
86          String firstFinalString = fillRestOfString(currentIndex + 1, firstText,
            stringBuilder1);
87          String secondFinalString = fillRestOfString(currentIndex + 1, secondText,
            stringBuilder2);
88
89          finalArrayList.add(firstFinalString);
90          finalArrayList.add(secondFinalString);
91
92          return finalArrayList;
93      }
94
95
96      //position one over and type
97      public static ArrayList<String> overAndIdentityTest(String text1, String text2) {
98          StringBuilder stringBuilder1 = new StringBuilder();
99          StringBuilder stringBuilder2 = new StringBuilder();
100         int currentIndex = -1;
101
102         ArrayList<String> finalArrayList = new ArrayList<>();
103
104         ArrayList<String> firstText = convertStringToArray(text1);
105         ArrayList<String> secondText = convertStringToArray(text2);
106
107         for (int i = 0; i < firstText.size(); i++) {
108             //tester
109             try {
110
111                 if (firstText.get(i).equals(secondText.get(i+1))) {
112                     stringBuilder1.append(firstText.get(i));
113                     currentIndex = i;
114                     break;
115
116                 } else if(secondText.get(i).equals(firstText.get(i+1))) {
117                     stringBuilder2.append(secondText.get(i));
118                     currentIndex = i;
119                     break;
120                 }
121
122                 stringBuilder1.append(firstText.get(i));
123                 stringBuilder2.append(secondText.get(i));
124
125             } catch (IndexOutOfBoundsException e) {
126                 break;
127             }
128
129             currentIndex = i;
130         }
131
132         String firstFinalString = fillRestOfString(currentIndex + 1, firstText,
            stringBuilder1);
133         String secondFinalString = fillRestOfString(currentIndex + 1, secondText,
```

```
            stringBuilder2);

            finalArrayList.add(firstFinalString);
            finalArrayList.add(secondFinalString);

            return finalArrayList;
        }



        //fills up rest of string
        public static String fillRestOfString(int i, ArrayList<String> text, StringBuilder
        stringBuilder) {

            for(int x = i; x < text.size(); x++) {
                stringBuilder.append(text.get(x));
            }

            return stringBuilder.toString();
        }

        //convert string into array list
        public static ArrayList<String> convertStringToArray(String text) {
            ArrayList<String> listOfStrings = new ArrayList<>();

            for (int i = 0; i < text.length(); i++) {
                listOfStrings.add(text.substring(i, i + 1));
            }

            return listOfStrings;
        }


        //ASF Calculations
        public static int ASFCalculations(String firstText, String secondText) {
            int finalValue = 0;
            int counter = 0;
            List<Character> charArray1 = new ArrayList<>();
            List<Character> charArray2 = new ArrayList<>();
            String longerText;
            String shorterText;

            if (firstText.length() >= secondText.length()) {
                longerText = firstText;
                shorterText = secondText;

            } else {

                longerText = secondText;
                shorterText = firstText;

            }

            for (int i = 0; i < firstText.length(); i++) {
                charArray1.add(firstText.charAt(i));
            }
            for (int x = 0; x < secondText.length(); x++) {
                charArray2.add(secondText.charAt(x));
            }

            for (int t = 0; t < shorterText.length(); t++) {
                finalValue = finalValue + (((int) charArray1.get(t) - 64) - ((int)
                charArray2.get(t) - 64));
            }

            for (int y = shorterText.length(); y < longerText.length(); y++) {
                counter++;
            }
```

```
200
201             finalValue = finalValue + counter;
202
203             return finalValue;
204
205
206         }
207
208
209     }
```