```java
//KrishJain
//ACSL Contest 2 2019 - 2020
//Junior Division
//String Differences

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.List;
import java.util.Scanner;

public class ACSLStringDiff_KrishJain_J_2020 {

    public final static List<String> VOWELS = Arrays.asList("A", "E", "I", "O", "U");

    public static void main(String[] args) {
        for (int i = 0; i <= 5; i++) {

            Scanner s = new Scanner(System.in);
            String input[] = s.nextLine().split(" ");

            String firstString = input[0];
            String secondString = input[1];

            List<String> firstStringCharacters = Arrays.asList(firstString.split(""));
            List<String> secondStringCharacters = Arrays.asList(secondString.split(""));

            List<String> firstStringRemovedDoubles =
            removeDoubles(firstStringCharacters);
            List<String> secondStringRemovedDoubles =
            removeDoubles(secondStringCharacters);

            List<String> firstStringRemovedVowels =
            removeVowels(firstStringRemovedDoubles);
            List<String> secondStringRemovedVowels =
            removeVowels(secondStringRemovedDoubles);

            List<String> firstStringUniqueCharactersLToR =
            removeLikeCharactersAtSamePositionsLToR(
                    firstStringRemovedVowels, secondStringRemovedVowels);
            List<String> secondStringUniqueCharactersLToR =
            removeLikeCharactersAtSamePositionsLToR(
                    secondStringRemovedVowels, firstStringRemovedVowels);

            List<String> firstStringUniqueCharactersRToL =
            removeLikeCharactersAtSamePositionsRToL(
                    firstStringUniqueCharactersLToR, secondStringUniqueCharactersLToR);
            List<String> secondStringUniqueCharactersRToL =
            removeLikeCharactersAtSamePositionsRToL(
                    secondStringUniqueCharactersLToR, firstStringUniqueCharactersLToR);

            String processedFirstString = String.join("",
            firstStringUniqueCharactersRToL);
            String processedSecondString = String.join("",
            secondStringUniqueCharactersRToL);

            if (processedFirstString.length() == processedSecondString.length()) {

                System.out.println(processedFirstString.compareTo(processedSecondString)
                <= 0 ? processedFirstString
                        : processedSecondString);
            } else {
                System.out.println(processedFirstString.length() <
                processedSecondString.length() ? processedFirstString
                        : processedSecondString);
            }
        }
    }
```

```
57
58      private static List<String> removeDoubles(List<String> stringCharacters) {
59          List<String> removedDoubles = new ArrayList<>();
60          for (int i = 0; i < stringCharacters.size(); i++) {
61              if (i == stringCharacters.size() - 1 ||
                !stringCharacters.get(i).equals(stringCharacters.get(i + 1))) {
62                  removedDoubles.add(stringCharacters.get(i));
63              }
64          }
65
66          return removedDoubles;
67      }
68
69      private static List<String> removeVowels(List<String> stringCharacters) {
70          List<String> stringCharactersWithoutFirstChar = stringCharacters.subList(1,
            stringCharacters.size());
71
72          stringCharactersWithoutFirstChar.removeAll(VOWELS);
73
74          stringCharactersWithoutFirstChar.add(0, stringCharacters.get(0));
75
76          return stringCharactersWithoutFirstChar;
77      }
78
79      private static List<String> removeLikeCharactersAtSamePositionsLToR(List<String>
        removeFrom,
80              List<String> comparingTo) {
81          int shorterStringSize = Math.min(removeFrom.size(), comparingTo.size());
82
83          List<String> stringUniqueCharacters = new ArrayList<>();
84          for (int i = 0; i < shorterStringSize; i++) {
85              if (!removeFrom.get(i).equals(comparingTo.get(i))) {
86                  stringUniqueCharacters.add(removeFrom.get(i));
87              }
88          }
89
90          if (shorterStringSize < removeFrom.size()) {
91              stringUniqueCharacters.addAll(removeFrom.subList(shorterStringSize,
                removeFrom.size()));
92          }
93
94          return stringUniqueCharacters;
95      }
96
97      private static List<String> removeLikeCharactersAtSamePositionsRToL(List<String>
        removeFrom,
98              List<String> comparingTo) {
99          int shorterStringSize = Math.min(removeFrom.size(), comparingTo.size());
100
101         List<String> stringUniqueCharacters = new ArrayList<>();
102         for (int i = 0; i < shorterStringSize; i++) {
103             if (!removeFrom.get(removeFrom.size() - 1 -
                i).equals(comparingTo.get(comparingTo.size() - 1 - i))) {
104                 stringUniqueCharacters.add(0, removeFrom.get(removeFrom.size() - 1 - i));
105             }
106         }
107
108         if (shorterStringSize < removeFrom.size()) {
109             stringUniqueCharacters.addAll(0, removeFrom.subList(0, shorterStringSize));
110         }
111
112         return stringUniqueCharacters;
113     }
114 }
```