**Name Matching Problem**(100 Marks)

The serious problem we are facing is the account opening fraud. In this growing type of fraud, criminals open accounts at financial institutions only to one day max out credit lines and cash advances before disappearing.

To avoid this, the institutions employ the screening process at the entry level. An exhaustive list of illegitimate and fraudulent account holders is maintained centrally which is considered as defaulting customers. The institution then will like to check every new account holder’s detail with the defaulting list and will ensure that the new account holder name is not matching with defaulting list.

Hence, the problem is to perform name matching. The problem is to match name 1 and name 2 using variety of logic.

Please check the following test cases for more understanding.

|  |
| --- |
| **Example 1** |
| Modi Naitik |
| Mr. Naitik Modi |
| Match |

It will be expected to remove all the special words like Mr., Mrs. Ms., Shri., etc. from both name 1 and name 2. Similarly, the special characters can also be removed.

In example 1, it is simply the permutation of the words. Hence, Naitik Modi is matching with Modi Naitik.

|  |
| --- |
| **Example 2** |
| sumayya abou |
| SUMAYYABANU |
| Match |

It is visible from example 2, that the matching should not be case sensitive. Also, these two names are matched with fuzzy matching logic where the threshold for matching is expected to be 80% and above.

|  |
| --- |
| **Example 3** |
| Shyama Prasad Khumawat |
| Shyama Khumawat |
| Match |

The above example specifies even if the subset matches from one string to the other, it is a match.

|  |
| --- |
| **Example 4** |
| SANKAR CHARAN DAS |
| s c das |
| Match |

In this, the two strings are matched as their initials are matching. It is expected to match initials in any order. Hence Sankar Charan Das should also match with “c s das” or “das c s”.

|  |
| --- |
| **Example 5** |
| Mrs Meera Sakhare |
| Mira Sakhare |
| Match |

This is example for phonetic matching. Two names sounding same but spelled differently should also match.

|  |
| --- |
| **Example 6** |
| Sureshkumar Periyal |
| Suresh Kumar Periyal |
| Match |

In this example, two names are same, but they are bonded in different way.

Along with positive matching, there will be some test cases which are ‘No Match’. If two names are not getting matched with all the possible logic applied, then they are ‘No Match’. Please check the following examples for more clarity.

|  |
| --- |
| **Example 7** |
| Shri. Veer Prasad Lodha |
| Virchandra Prasad |
| No Match |

|  |
| --- |
| **Example 8** |
| Radhika Sharma |
| Mrs. Sharma Kaveri |
| No Match |

**Input Format**

The first line of input consists of Name 1.  
The second line of input consist of Name 2.

**Constraints**

1<= Name 1, Name 2 <=50

**Output Format**

Print Matchif the names are a match otherwise print No Match.

Sample TestCase 1

Input

Modi Naitik

Mr. Naitik Modi

Output

Match

**Time Limit(X):**

1.00 sec(s) for each input.

**Memory Limit:**

512 MB

**Source Limit:**

100 KB

**Allowed Languages:**

C, C++, C++11, C++14, C#, Java, Java 8, Kotlin, PHP, PHP 7, Python, Python 3, Perl, Ruby, Node Js, Scala, Clojure, Haskell, Lua, Erlang, Swift, VBnet, Js, Objc, Pascal, Go, F#, D, Groovy, Tcl, Ocaml, Smalltalk, Cobol, Racket, Bash, GNU Octave, Rust, Common LISP, R, Julia, Fortran, Ada, Prolog, Icon, Elixir, CoffeeScript, Brainfuck, Pypy, Lolcode, Nim, Picolisp, Pike, pypy3

Soln:

/\* Read input from STDIN. Print your output to STDOUT\*/

import java.io.\*;

import java.util.\*;

public class CandidateCode {

static ArrayList<String> removeSpecialWords(String name){

String[] arr = name.split(" ");

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

int i=0;

for(String s: arr){

if(s.equalsIgnoreCase("Mr.") || s.equalsIgnoreCase("Ms.") || s.equalsIgnoreCase("Mrs.") || s.equalsIgnoreCase("Shri."))

continue;

else{

res.add(s.toLowerCase());

i++;

}

}

return res;

}

static Set<String> isEqual(ArrayList<String> n1, ArrayList<String> n2){

Set<String> nameSet = new HashSet<String>();

nameSet.addAll(n1);

nameSet.addAll(n2);

return nameSet;

}

static Set<String> oneWordDiffSpell(Set<String> nameSet){

String sameName="";

for(String s: nameSet){

if(s.contains("ee")){

sameName = s.replace("ee","i");

if(nameSet.contains(sameName)){

nameSet.remove(sameName);

break;

}

}

else if(s.contains("ph")){

sameName = s.replace("ph","f");

if(nameSet.contains(sameName)){

nameSet.remove(sameName);

break;

}

}

else if(s.contains("ie")){

sameName = s.replace("ie","y");

if(nameSet.contains(sameName)){

nameSet.remove(sameName);

break;

}

}

else if(s.contains("ie")){

sameName = s.replace("ie","ee");

if(nameSet.contains(sameName)){

nameSet.remove(sameName);

break;

}

}

else if(s.contains("ksh")){

sameName = s.replace("ksh","x");

if(nameSet.contains(sameName)){

nameSet.remove(sameName);

break;

}

}

}

return nameSet;

}

static Set<String> initialsPresent(Set<String> nameSet){

int singleEle=0;

while(true){

for(String s : nameSet){

if(s.length()>2){

String str=""+s.charAt(0);

if(nameSet.contains(str)){

nameSet.remove(str);

singleEle-=1;

break;

}

}

else{

singleEle+=1;

}

}

if(singleEle!=0){

singleEle=0;

}

else if(singleEle==0)break;

}

return nameSet;

}

static boolean endProgram(Set<String> nameSet,ArrayList<String> nameArr1, ArrayList<String> nameArr2){

if(nameSet.size() == nameArr1.size() || nameSet.size() == nameArr2.size()){

return true;

}

return false;

}

static boolean diffBonded(ArrayList<String> nameArr1, ArrayList<String> nameArr2){

String str1="";

String str2="";

for(String s: nameArr1){

str1+=s;

}

for(String s: nameArr2){

str2+=s;

}

if(str1.equals(str2))return true;

//else return diffLogic(str1,str2);

return false;

}

static boolean diffLogic(String str1,String str2){

int len,i;

int isPart=0,isNotPart1=0,isNotPart2=0;

char c1,c2;

if(str1.length()>=str2.length())len=str1.length();

else len=str2.length();

for(i=0;i<len;i++){

if(i<str1.length())

c1=str1.charAt(i);

else{

isNotPart2+=str2.length()-i+1;

break;

}

if(i<str2.length())

c2=str2.charAt(i);

else{

isNotPart1+=str1.length()-i+1;

break;

}

if(c1==c2)isPart++;

else{

isNotPart1++;

isNotPart2++;

}

}

//i++;

//isNotPart1=str1.length()-i;

//isNotPart2=str2.length()-i;

if(isPart>isNotPart1 || isPart>isNotPart2)

return true;

return false;

}

public static void main(String args[] ) throws Exception {

String name1,name2;

Scanner in = new Scanner(System.in);

name1=in.nextLine();

name2=in.nextLine();

ArrayList nameArr1 = removeSpecialWords(name1);

ArrayList nameArr2 = removeSpecialWords(name2);

boolean endProgram=false;

Set<String> nameSet = isEqual(nameArr1,nameArr2);

if(endProgram(nameSet,nameArr1,nameArr2)){

System.out.println("Match");

}

else {

nameSet = initialsPresent(nameSet);

if(endProgram(nameSet,nameArr1,nameArr2)){

System.out.println("Match");

}

else{

nameSet = oneWordDiffSpell(nameSet);

if(endProgram(nameSet,nameArr1,nameArr2)){

System.out.println("Match");

}

else{

if(diffBonded(nameArr1,nameArr2)){

System.out.println("Match");

}

else{

if(diffLogic(name1.toLowerCase(),name2.toLowerCase())){

System.out.println("Match");

}

else System.out.println("No Match");

}

}

}

}

}

}







