Asian Pacific Heritage Coding Contest

As part of a coding contest being organized to honor Asian Pacific American Heritage Month, solve this simple string manipulation challenge. Given a pair of strings, make them equal by doing **at most** one of the following operations **at most once**:

* Select two characters, let’s say *char1* and *char2,*and replace all occurrences of *char1 in the first string with char2.*
* Select two characters, let’s say *char1* and *char2,*and replace all occurrences of *char1 in the second string with char2.*

**For example,** the string “aabc” can be converted to any of the following strings by performing the aforementioned operation: “bbbc”, “eebc”, “aacc”, “aadc”, etc. However, it cannot be converted to the string “bcbc” (Since all the occurrences of a character can be replaced by a single new character).

**Note** that doing both the operations simultaneously is not allowed. For example, given two strings “ac” and “ca”, we can not convert both the a’s in the respective strings to ‘c’ to make them equal.

Given two arrays of n strings each, compare the strings present at the same indices in both the arrays and find if they can be made equal or not. Return an array of n strings, where the ith element would be YES if the strings at index i can be equalized, otherwise, the answer is NO.

**Example**

Consider the array of strings to be:

* firstStrings = [“aaa”, “abbc”, “zyz”]
* secondStrings = [“bbb”, “cccc”, “zyx”]

Let us compare the strings at each index:

* i = 0: firstStrings[0] = “aaa”, secondStrings[0] = “bbb”

We can change all the occurrences of ‘a’ to ‘b’ in the first string in this case. Therefore, the answer is “YES”.

* i = 1: firstStrings[1] = “abbc”, secondStrings[1] = “cccc”

There is no way in which we can make these strings identical. Therefore, the answer is “NO”.

* i = 2: firstStrings[2] = “zyz”, secondStrings[2] = “zyx”

We can replace the only occurrence of ‘x’ in the second string with ‘z’ to make them identical. Therefore, the answer is “YES”.

Hence, the answer corresponding to each pair of strings is [“YES”, “NO”, “YES”].

**Function Description**

Complete the function canBeEqualized in the editor below.

canBeEqualized has the following parameters:

firstStrings[firstStrings[0],…firstStrings[n-1]]:  an array of strings representing the first string of each pair

secondStrings[secondStrings[0],…secondStrings[n-1]]:  an array of strings representing the second string of each pair

**Returns**

    string[]: an array of strings representing the answer to each pair of strings

**Constraints**

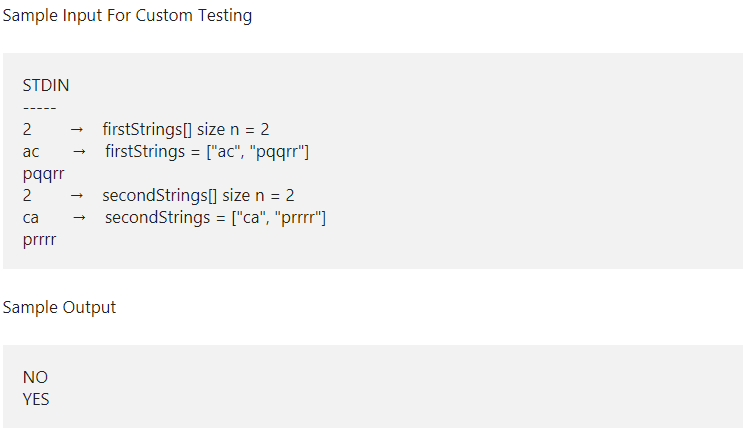
* 1 ≤ n ≤ 103
* 1 ≤ |firstStringsi|, |secondStringsi| ≤ 103
* All the strings consist of lowercase English characters only.

### **Input Format For Custom Testing**

The first line contains an integer, n, denoting the number of elements in firstStrings.  
Each line i of the n subsequent lines (where 0 ≤ i < n) contains a string describing firstStringsi.

The next line contains an integer, n, denoting the number of elements in secondStrings.  
Each line i of the n subsequent lines (where 0 ≤ i < n) contains a string describing secondStringsi.

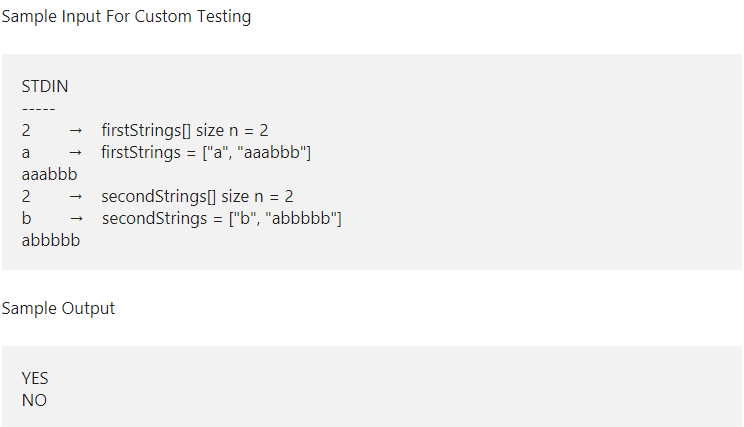
### **Sample Case 0**



Explanation

* For the first pair of strings, we can check that there is no way to make them equal, thus the answer is “NO”.
* For the second pair of strings, we can convert all the occurrences of ‘q’ in the first string to ‘r’, thus the answer is “YES”.

### **Sample Case 1**



Explanation

* For the first pair of strings, we can convert the only occurrence of ‘a’ to ‘b’, thus the answer is “YES”.
* For the second pair of strings, it can be seen that it is impossible to make the two strings equal, thus the answer is “NO”.