



Problem H
Anagrammatic Distance

Input File: H.IN

Output File: standard output

Program Source File: H.C, H.CPP, H.JAVA, H.PAS

Two words are said to be *anagrams of each other* if the letters from one word can be rearranged to form the other word. For example, *occurs* is an anagram of *succor*; however, *dear* is not an anagram of *dared* (because the *d* appears twice in *dared*, but only once in *dear*). The most famous anagram pair (in English) is *dog* and *god*.

The *anagrammatic distance* between any two words is the minimum number of letters which must be removed so that the remaining portions of the two words become anagrams. For example, given the words *sleep* and *leap*, we need to remove a minimum of three letters --- two from *sleep* and one from *leap* --- before what's left are anagrams of each other (in each case, *lep*). With words such as *dog* and *cat*, where the two have absolutely no letters in common, the anagrammatic distance is an extreme (explicitly 6) since all the letters need to be removed. (Here, a word is always an anagram of itself.)

You must write a program to calculate the anagrammatic distance between any two given words.

Input:

The first line of the input will contain a positive integer value *N* (less than 60,000) indicating the number of cases. Each case will consist of two words, possibly empty, each given on a single line (for a total of $2N$ additional lines).

Although they may have zero length, the words are simple --- the letters are all lowercase and are taken from the usual twenty-six letter English alphabet (*abcdefghijklmnopqrstuvwxyz*). The longest word is *pneumonoultramicroscopicsilicovolcanoconiosis*.

Output:

The output should consist of the case number and the anagrammatic distance, formatted as shown.

Sample Input:	Sample Output
4	Case #1: 0
crocus	Case #2: 1
succor	Case #3: 5
dares	Case #4: 4
seared	
empty	
smell	
lemon	