Making Anagrams

We consider two strings to be anagrams of each other if the first string's letters can be rearranged to form the second string. In other words, both strings must contain the same exact letters in the same exact frequency. For example, `bacdc` and `dcbac` are anagrams, but `bacdc` and `dcbad` are not.

Alice is taking a cryptography class and finding *anagrams* to be very useful. She decides on an encryption scheme involving two large strings where encryption is dependent on the minimum number of character deletions required to make the two strings anagrams. Can you help her find this number?

Given two strings, $s1$ and $s2$, that may not be of the same length, determine the minimum number of character deletions required to make $s1$ and $s2$ anagrams. Any characters can be deleted from either of the strings.

Example.
$s1 = \mathbf{abc}$
$s2 = \mathbf{amnop}$

The only characters that match are the $\mathbf{a}$'s so we have to remove $\mathbf{bc}$ from $s1$ and $\mathbf{mnop}$ from $s2$ for a total of 6 deletions.

Function Description

Complete the *makingAnagrams* function in the editor below.

makingAnagrams has the following parameter(s):

- *string s1:* a string
- *string s2:* a string

Returns

- *int:* the minimum number of deletions needed

Input Format

The first line contains a single string, $s1$.
The second line contains a single string, $s2$.

Constraints

- $1 \leq |s1|, |s2| \leq 10^4$
- It is guaranteed that $s1$ and $s2$ consist of lowercase English letters, ascii[a-z].

Sample Input

```
cde
abc
```

Sample Output

```
4
```

Explanation

Delete the following characters from our two strings to turn them into anagrams:

1. Remove `d` and `e` from `cde` to get `c`.
2. Remove `a` and `b` from `abc` to get `c`.

4 characters have to be deleted to make both strings anagrams.