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.

For example, $s1 = \mathbf{abc}$ and $s2 = \mathbf{amnop}$. 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. It should return an integer representing the minimum number of deletions needed to make the strings anagrams.

makingAnagrams has the following parameter(s):

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

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].

Output Format

Print a single integer denoting the minimum number of characters which must be deleted to make the two strings anagrams of each other.

Sample Input

```
cde
abc
```

Sample Output

```
4
```

Explanation

We delete the following characters from our two strings to turn them into anagrams of each other:

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

We had to delete 4 characters to make both strings anagrams.