You found your very first laptop in the attic, and decided to give in to nostalgia and turn it on. The laptop turned out to be password protected, but you know how to crack it: you have always used the same password, but encrypt it using *permutation ciphers* with various keys. The key to the cipher used to protect your old laptop very conveniently happened to be written on the laptop lid.

Here's how *permutation cipher* works: the key to it consists of all the letters of the alphabet written up in some order. All occurrences of letter 'a' in the encrypted text are substituted with the first letter of the key, all occurrences of letter 'b' are replaced with the second letter from the key, and so on, up to letter 'z' replaced with the last symbol of the key.

Given the password you always use, your task is to encrypt it using the *permutation cipher* with the given key.

Example

For password = "iamthebest" and  
key = "zabcdefghijklmnopqrstuvwxy", the output should be  
permutationCipher(password, key) = "hzlsgdadrs".

Here's a table that can be used to encrypt the text:

abcdefghijklmnopqrstuvwxyz

|| | || | ||

vv v vv v vv

zabcdefghijklmnopqrstuvwxy

Input/Output

* **[execution time limit] 4 seconds (py3)**
* **[input] string password**

The password you always use. It is guaranteed to consist only of lowercase English letters.  
If you're using Python 2, note that the string is given as a unicode.

*Guaranteed constraints:*  
1 ≤ password.length ≤ 26.

* **[input] string key**

The key to the *permutation cipher*. It is guaranteed to be a permutation of the [plaintext alphabet](keyword://plaintext-alphabet).  
If you're using Python 2, note that the string is given as a unicode.

*Guaranteed constraints:*  
key.length = 26.

* **[output] string**
  + Your password encrypted by the *permutations cipher* with the given key.
* def permutationCipher(password, key):
* table = str.maketrans("abcdefghijklmnopqrstuvwxyz",key)
* return password.translate(table)