726 Decode

Any one-to-one mapping f of the alphabet to itself can be used to encode text by replacing each occurence of letter `c` with f(`c`).

Example: One such mapping could be the mapping of a letter to three positions beyond the letter in the alphabet: For example, `a->d`, `b->e`, `c->f`, `d->g`, ... etc.

natural language: `The car is blue`.
encoded message: `Wkh fdu lv eoxh`.

The mapping that defines the encoding is one-to-one. That is, two different letters do not map to the same letter of the alphabet (`a->x` and `t->x` is impossible).

Input and Output

The input file contains two messages separated by a blank line. We will refer to them as '*KNOWN*' and '*ENCODED*' messages. This file contains an encoded message.

You are to write a program that decodes the message in the file according to the following guidelines.

1. The input message *KNOWN* contains a natural language text that is written by the same person that wrote the encoded message in *ENCODED*. It is to be assumed that the person uses letters of the alphabet with the same relative frequency from document to document. Read *KNOWN* and count the number of times each letter of the alphabet occurs. Order the letters of the alphabet in such a way that the first in the ordering is the most frequently used letter in *KNOWN* and the last in the list is the least frequently used letter in *KNOWN*. Break all ties for letters that occur at the same frequency alphabetically. That is, if both `p` and `w` appeared in *KNOWN* exactly 12 times each, then `p` would precede `w` in the ordered list.
2. Read *ENCODED* and count the number of times each encoded letter of the alphabet occurs. Order the letters in such a way that the first in the ordering is the most frequently used letter in *ENCODED* and the last in the list is the least frequently used letter in *ENCODED*. Again, break all ties alphabetically.
3. The process for counting the frequencies with which letters occur is to be case insensitive. That is, the appearance of the letter `W` followed by `w` would account for two occurences of that letter.
4. The counting process is only to count letters of the alphabet, not other characters.
5. The program is to assume that the most frequently used letter in *KNOWN* maps to the most frequently used letter in *ENCODED*; the second most frequently used letter in *KNOWN* maps to the second most frequently used letter in *ENCODED*; ... etc.
6. Your program is to use knowledge of the mapping described in specification 5 to decode the message and put the decoded message in the output file.
7. Even though the counting process was to be case insensitive, the deconding process is to be case sensitive. That is, encoded capital letters are to be decoded into capital letters, and encoded lower case letters are to be decoded into lower case letters.

8. All non-alphabetic characters found in *ENCODED* are to pass unchanged and appear in the same relative position in the output file as in the input message *ENCODED*. For example, this message may contain blank lines which must appear also in the output file.

Sample Input

```
The car is blue.

Wkh fdu lv eoxh.
```

Sample Output

```
The car is blue.
```