CCC '06 S2 - Attack of the CipherTexts

2006 Canadian Computing Competition, Stage 1

Ruby is a code-breaker. She knows that the very bad people (Mr. X and Mr. Z) are sending secret messages about very bad things to each other.

However, Ruby has managed to intercept a plaintext message and the corresponding ciphertext message. By plaintext, we mean the message before it was encrypted (i.e., readable English sentences), and by ciphertext, we mean the message after it was encrypted (i.e., gibberish). To encrypt a message, each letter is changed to a new letter, so that if you read the ciphertext message, it is not obvious what the plaintext message is.

However, Ruby being the outstanding code-breaker she is, knows the algorithm that Mr. X and Mr. Z use. She knows they simply map one letter to another (possibly different) letter when they encrypt their messages. Of course, this map must be "one-to-one", meaning that each plaintext letter must correspond to exactly one ciphertext letter, as well as "onto", meaning that each ciphertext letter has exactly one corresponding plaintext letter.

Your job is to automate Ruby's codebreaking and help save the world.

Input Specification

The input consists of 3 strings, with each string on a separate line. The first string is the plaintext message which Ruby knows about. The second string is the ciphertext message which corresponds to the plaintext message. The third string is another ciphertext message. You may assume that all strings have length of at least 1 character and at most 80 characters. You can also assume that there are only 27 valid characters: the uppercase letters (`A` through `Z`) as well as the space character (` `). That is, there will be no punctuation, lowercase letters, or special characters (like `!` or `@`) in either the plaintext or ciphertext messages.

Output Specification

The output is a (plaintext) string which corresponds to the second ciphertext input. It may not be possible to determine each character of the second ciphertext string, however. If this is the case, the output should have a period (`.`) character for those letters which cannot be determined.

Sample Input 1

```
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
UIFARVJDLACSPXOAGPYAKVNQTAPWFSAUIFAMB ZAEPH
XFABSFAWFSZACBEAQFPQMFAEPJOHAWFSZACBEAUIJOHTAIBAIB
```

Sample Output 1

```
WE ARE VERY BAD PEOPLE DOING VERY BAD THINGS HA HA
```

Explanation for Sample Output 1

Notice that every plaintext character is in the first message, and so, the entire mapping can be computed.

Sample Input 2

```
THERE ARE NOT ENOUGH LETTERS
XQAZASEZASNYXSANYLWQSTAXXAZM
JSCENNYXSIACYIASXQJM
```

Sample Output 2

```
. .ANNOT .E.O.E TH.S
```

Explanation for Sample Output 2

Notice that the characters that cannot be determined are the (ciphertext) letters `J`, `C`, `I`, since these ciphertext letters do not appear in the earlier ciphertext message. It turns out that the plaintext message for the last ciphertext was `I CANNOT DECODE THIS`. All other letters should correspond between plaintext and ciphertext.

CCC problem statements in large part from the PEG OJ