**[Substitution](https://cs50.harvard.edu/x/2020/psets/2/substitution/" \l "substitution)**

Implement a program that implements a substitution cipher, per the below.

$ ./substitution JTREKYAVOGDXPSNCUIZLFBMWHQ

plaintext: HELLO

ciphertext: VKXXN

[**Background**](https://cs50.harvard.edu/x/2020/psets/2/substitution/#background)

In a substitution cipher, we “encrypt” (i.e., conceal in a reversible way) a message by replacing every letter with another letter. To do so, we use a *key*: in this case, a mapping of each of the letters of the alphabet to the letter it should correspond to when we encrypt it. To “decrypt” the message, the receiver of the message would need to know the key, so that they can reverse the process: translating the encrypt text (generally called *ciphertext*) back into the original message (generally called *plaintext*).

A key, for example, might be the string NQXPOMAFTRHLZGECYJIUWSKDVB. This 26-character key means that A (the first letter of the alphabet) should be converted into N (the first character of the key), B (the second letter of the alphabet) should be converted into Q (the second character of the key), and so forth.

A message like HELLO, then, would be encrypted as FOLLE, replacing each of the letters according to the mapping determined by the key.

Let’s write a program called substitution that enables you to encrypt messages using a substitution cipher. At the time the user executes the program, they should decide, by providing a command-line argument, on what the key should be in the secret message they’ll provide at runtime.

Here are a few examples of how the program might work. For example, if the user inputs a key of YTNSHKVEFXRBAUQZCLWDMIPGJO and a plaintext of HELLO:

$ ./substitution YTNSHKVEFXRBAUQZCLWDMIPGJO

plaintext: HELLO

ciphertext: EHBBQ

Here’s how the program might work if the user provides a key of VCHPRZGJNTLSKFBDQWAXEUYMOI and a plaintext of hello, world:

$ ./substitution VCHPRZGJNTLSKFBDQWAXEUYMOI

plaintext: hello, world

ciphertext: jrssb, ybwsp

Notice that neither the comma nor the space were substituted by the cipher. Only substitute alphabetical characters! Notice, too, that the case of the original message has been preserved. Lowercase letters remain lowercase, and uppercase letters remain uppercase.

Whether the characters in the key itself are uppercase or lowercase doesn’t matter. A key of VCHPRZGJNTLSKFBDQWAXEUYMOI is functionally identical to a key of vchprzgjntlskfbdqwaxeuymoi (as is, for that matter, VcHpRzGjNtLsKfBdQwAxEuYmOi).

And what if a user doesn’t provide a valid key?

$ ./substitution ABC

Key must contain 26 characters.

Or really doesn’t cooperate?

$ ./substitution

Usage: ./substitution key

Or even…

$ ./substitution 1 2 3

Usage: ./substitution key