Table of Contents

Vigenère

tl;dr

Implement a program that encrypts messages using Vigenère's cipher, per the below.

```
$ ./vigenere ABC
plaintext:  HELLO
ciphertext: HFNLP
```

Background

Vigenère's cipher improves upon Caesar's cipher (../caesar/caesar.html) by encrypting messages using a sequence of keys (or, put another way, a keyword). In other words, if *p* is some plaintext and *k* is a keyword (i.e., an alphbetical string, whereby A represents 0, B represents 1, C represents 2, …, and Z represents 25), then each letter, c_i, in the ciphertext, *c*, is computed as:

$$c_i = (p_i + k_j) \bmod 26$$

Note this cipher's use of k_j as opposed to just *k*. And if *k* is shorter than *p*, then the letters in *k* must be reused cyclically as many times as it takes to encrypt *p*.

In other words, if Vigenère himself wanted to say HELLO to someone confidentially, using a keyword of, say, ABC, he would encrypt the H with a key of 0 (i.e., A), the E with a key of 1 (i.e., B), and the first L with a key of 2 (i.e., C), at which point he'd be out of letters in the keyword, and so he'd reuse (part of) it to encrypt the second L with a key of 0 (i.e., A) again, and the O with a key of 1 (i.e., B) again. And so he'd write HELLO as HFNLP.

Table 1. Encrypting HELLO with a keyword of ABC (reused cyclically as ABCAB) yields HFNLP.

plaintext	H	E	L	L	O
	A	B	C	A	B
+ key					
	0	1	2	0	1
= ciphertext	H	F	N	L	P

Specification

Design and implement a program that encrypts messages using Vigenère's cipher.

- Implement your program in a file called `vigenere.c` in a directory called `vigenere`.
- Your program must accept a single command-line argument: a keyword, *k*, composed entirely of alphabetical characters.
- If your program is executed without any command-line arguments, with more than one command-line argument, or with one command-line argument that contains any non-alphabetical character, your program should print an error (of your choice) and exit immediately, with `main` returning `1` (thereby signifying an error).
- Otherwise, your program must proceed to prompt the user for a string of plaintext, *p*, (as by a prompt for `plaintext:`) which it must then encrypt according to Vigenère's cipher with *k*, ultimately printing the result (prepended with `ciphertext:` and ending with a newline) and exiting, with `main` returning `0`.
- With respect to the characters in *k*, you must treat `A` and `a` as 0, `B` and `b` as 1, … , and `Z` and `z` as 25.
- Your program must only apply Vigenère's cipher to a character in *p* if that character is a letter. All other characters (numbers, symbols, spaces, punctuation marks, etc.) must be outputted unchanged. Moreover, if your code is about to apply the j^{th} character of *k* to the i^{th} character of *p*, but the latter

proves to be a non-alphabetical character, you must wait to apply that j^{th} character of *k* to the next alphabetical character in *p*; you must not yet advance to the next character in *k*.

- Your program must preserve the case of each letter in *p*.

Walkthrough

Usage

Your program should behave per the examples below. Assumed that the underlined text is what some user has typed.

```
$ ./vigenere 13
Usage: ./vigenere k
```

```
$ ./vigenere
Usage: ./vigenere k
```

```
$ ./vigenere bacon and eggs
Usage: ./vigenere k
```

```
$ ./vigenere bacon
plaintext: Meet me at the park at eleven am
ciphertext: Negh zf av huf pcfx bt gzrwep oz
```

Testing

To help you test `vigenere`, we've written a program called `devigenere` for you that also takes one and only one command-line argument (a keyword) but whose job is to take ciphertext as input and produce plaintext as output. To use our program, execute

```
~cs50/pset2/devigenere k
```

at your prompt, where `k` is some keyword. Presumably you'll want to paste your program's output as input to our program; be sure, of course, to use the same key. Note that you do not need to implement `devigenere` yourself, only `vigenere`.

Correctness

```
check50 cs50/2018/x/vigenere
```

Style

```
style50 vigenere.c
```

Staff's Solution

```
~cs50/pset2/vigenere
```

Hints

Not sure where to begin? As luck would have it, this program's pretty similar to `caesar` (../caesar/caesar.html)! Only this time, you need to decide which character in *k* to use as you iterate from character to character in *p*.