CS 215 - Fundamentals of Programming II
Fall 2019 - Homework 1
10 points

Out: August 28, 2019
Due: September 4, 2019 (Wednesday), by 11:59pm

The purpose of this homework is to practicing using the development environment expected in this course. Therefore, please type in your program from scratch in a text editor and use the clang++ compiler for this homework. **Please ask for assistance sooner rather than later if you need it.**

Problem Statement
A *cipher* is a form of cryptography in which individual letters of a message are replaced by other letters or symbols. The original form of the message is called the *plaintext*. It is enciphered using the cipher into the *ciphertext*. The act of recovering the plaintext is called *deciphering*.

A very simple cipher was used by the Romans and is now called the Caesar shift cipher. In the Caesar shift cipher, an alphabetic character (i.e., a letter) is replaced by the letter located *n* places to the right in the alphabet circularly. Another way to say this is that the plaintext letter is shifted circularly to the right *n* places to obtain the ciphertext letter. For example, if the shift is 8, then 'A' becomes 'I', 'B' becomes 'J', etc. to 'Z' becomes 'H' This idea also can be represented as a substitution table, as shown below for a shift of 8.

Plaintext	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Ciphertext	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H

Using the corrected file copying program from Lectures 3 and 4 as a base, modify it to perform a Caesar shift encipherment of **only** the alphabetic characters in the input file before writing the (enciphered) character to the output file. The source code for this program must be in a file named **`cipher.cpp`**.

Coding Notes
This assignment is being distributed by the GitKeeper submission system. You should have received an email with the URL to clone. A reminder of the commands needed to set up this assignment:

```
$ cd <path to your class directory>
$ git clone <URL given in the assignment email>
```

This will create a working directory called **`hwk01-cipher`** with a sample plaintext file. Change to this directory and create **`cipher.cpp`**.

```
$ cd hwk01-cipher
$ emacs cipher.cpp &
```

Be sure to commit your changes early and often. **The first commit for this assignment is expected to be by Friday, August 30.** A reminder of the commands needed to do this (run in the working directory):

```
$ git add .
$ git commit -m "<description of what was completed>"
```

The encipherment must be **case sensitive**. That is, an uppercase letter must map to another uppercase letter (e.g., '`A`' to '`I`'), and a lowercase letter must map to another lowercase letter (e.g., '`a`' to '`i`'). The non-alphabetic characters (i.e., digits, punctuation, and whitespace) must be passed through as plaintext.

The program must accept exactly **three** command-line arguments that are the name of an input file, the name of an output file, and an integer. The input file is interpreted a plaintext message. The output file will contain the corresponding ciphertext. The integer is the shift distance. **The program must do proper error checking of the number of command line arguments and the file opens.** Here are some things to note:

- Since `argv` is an array of C-strings, the third command-line argument will need to be converted to an integer. This can be done using the function `stoi` that is defined in the `<string>` library. It is used as follows:

  ```
  int shift = stoi(argv[3]);
  ```

  Note: this C++11 function is functionally equivalent to the C function `atoi`, but it does error checking and throws an exception if the string does not form a valid integer, causing the program to "crash" with an unhandled exception. The program file must be compiled with the `-std=c++11` option.

- To compute the right circular shift of a lowercase alphabetic character `ch`, we can use the following formula:

  ```
  ch = (((ch - 'a') + shift) % 26) + 'a';
  ```

  Computing the shift of an uppercase alphabetic character is similar using '`A`' instead of '`a`'.

- Side note: Deciphering a Caesar shift is the opposite of enciphering, but subtracting the shift value could result in a negative number. The equivalent positive shift value is 26 minus the shift value. This means that we can decipher any Caesar shift message in ciphertext with the same program by giving a shift value that is 26 minus the shift value used to encipher. E.g., suppose we encipher using a shift value of 8, then to decipher the resulting ciphertext, we run the program using the ciphertext as the input file and a shift value of 18.

A reminder of how to compile this program using clang++:

```
$ clang++ cipher.cpp -o cipher -Wall -std=c++11
```

The `-Wall` option is used to turn on all of the warning messages. Treat warning messages like error messages (i.e., they must be "fixed") unless you understand why it can be ignored. An example run of this program would look like:

```
$ ./cipher plaintext.txt ciphertext.txt 8
```

For file `plaintext.txt`, given in the assignment repository, containing the following "message":

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Then file `ciphertext.txt` should contain the following result:

```
ijklmnopqrstuvwxyzabcdefghIJKLMNOPQRSTUVWXYZABCDEFGH
```

Note: you can create other test files yourself using your text editor.

To decipher the above `ciphertext.txt`, you would run:

```
$ ./cipher ciphertext.txt newplaintext.txt 18
```

and `newplaintext.txt` would be exactly the same as the original `plaintext.txt`. You can check this by using the UNIX `diff` command:

```
$ diff plaintext.txt newplaintext.txt
```

When two files are exactly the same, the `diff` command produces no output.

How to submit

Please note that the automated submission system requires all files be named exactly as specified in an assignment (`cipher.cpp`, in this case), and also requires that the output of the program be exactly as expected including whitespace, capitalization, and spelling.

Assignments are submitted to GitKeeper by pushing committed changes.

```
$ git push
```

Only a clean repository (one where all changes have been committed) can be pushed. If a repository is not clean, git will say everything is up to date, and refuse to do the push. To see which files have been changed, but not committed, use command:

```
$ git status
```

You can also see all the log messages by using command:

```
$ git log
```

Note that the first commit was done by the GitKeeper system.