Extra Credit Homework

I've been working on the Hail Ceasar$_{(shift)}$! railroad

Due: December 6 by 11:59:59 PM **Assigned:** November 27, 2018

A Caesar shift is one of the oldest forms of encryption. It was originally implemented by using two strips of paper with the alphabet written on them. One of the strips was then shifted a certain number of spaces over. For example, a shift of 3 would change the letter 'A' to 'D', 'B' to 'E', and so on. 'Z' would shift back around to 'C'.
A Caesar shift known as a substitution cipher, aptly named because it substitutes one letter for another
A rail fence cipher is categorized as a transposition cipher, because it only re-arranges the existing letters of a message.
This assignment will allow the user to provide options for the necessary actions to be taken when launching the program (encode/decode), the size of the Caesar shift, and the name of the file that that contains the message (of arbitrary length). See the requirements for to see what this will look like.

Requirements:

- Name your source file `exc.cpp`
- You are free to use the string class or C-Strings as you see fit
- You must declare a minimum of two functions, one to encode and another to decode
  - You are free to declare more than two, but there must be at least one function that does encoding and at least one function that does decoding
- The message will be read in from a file
  - The length of the message is arbitrary
  - The message will be on a single line only
- The message must be converted to upper case
  - **ONLY** letters should be converted
- You will perform a Caesar shift, the size of which is specified when the program is executed
  - The shift can only be a positive integer
  - The highest allowable shift is 25
  - **ONLY** letters will be shifted
- You will also perform a two layer rail fence cipher

  - **ALL** characters will be transposed

- Options will be provided when the program is executed
  - The order of the options may be assumed to always be the same
  - The first is always encode/decode (-e/-d), the second is the size of the shift, and the final option is the name of the file to be read
- The input file will be overwritten
  - This means that when the original message is encoded, there will be no copy of it left
  - That file will contain only the encoded message when the program is finished executing
  - The reverse is true for decoding
- A sample run of your program that encodes a file named message using a shift of 7 should look like:

```
[WSUID@localhost Homeworks]$ ./extra -e 7 message
[WSUID@localhost Homeworks]$
```

- The following demonstrate what the contents of the file message would be before and after executing the program (*NOTE*: This file only contains a single line, it had to be split in order to fit on the page)
  - Before:

    ```
    That which is clearly known hath less terror than that which is
    but hinted at and guessed.
    ```

  - After:

    ```
    AH OJ ZJLYFRVUOA LZAYV OUAH OJ ZIAOUL AHKNLZKOADPOP SHS UD HOSZ
    LYYAH OADPOP B PAKH U BZL.
    ```

- A sample run of your program that decodes a file named message using a shift of 7 should look like:

```
[WSUID@localhost Homeworks]$ ./extra -d 7 message
[WSUID@localhost Homeworks]$
```

- The following demonstrate what the contents of the file message would be before and after executing the program (*NOTE*: This file only contains a single line, it had to be split in order to fit on the page)
  - Before:

    ```
    AH OJ ZJLYFRVUOA LZAYV OUAH OJ ZIAOUL AHKNLZKOADPOP SHS UD HOSZ
    LYYAH OADPOP B PAKH U BZL.
    ```

  - After:

```
THAT WHICH IS CLEARLY KNOWN HATH LESS TERROR THAN THAT WHICH IS
BUT HINTED AT AND GUESSED.
```

Hints:

- You will be writing to the same file you are reading from; make sure not to cross the streams
- Test each encoding method alone
- http://www.geeksforgeeks.org/command-line-arguments-in-c-cpp/
- https://www.cprogramming.com/tutorial/lesson14.html
- http://www.learncpp.com/cpp-tutorial/713-command-line-arguments/

Reminders:

- Be sure that your program includes your name, ID, description, etc. as shown in the General Homework Requirements Handout
- Use good style including indentation, comments, etc. Part of the grade will be for style and quality.
- Carefully test your program.
- You are welcome to write your program at home. If you do, be sure to compile and test it in the lab before submitting it.

How to submit your program:

- Submit the file `exc.cpp` electronically using the following terminal command:
  For the 12:30 lecture section:
  `~cs211a/bin/handin exc exc.cpp`

  For the 5:35 lecture section:
  `~cs211b/bin/handin exc exc.cpp`