Program 2 - Cryptography

For this assignment you will write a program that can encrypt and decrypt data using a variation of a Symmetric Key Cipher. All code must be compiled and run in CLion and must be submitted via GitHub by **6:00am CST, Monday February 22, 2021**. You may submit by **6:00am CST, Wednesday February 24, 2021** for a 30 point penalty.

Background

The practice of Encryption, or **Cryptogrophy**, involves converting an original message, or critical piece of information, from plaintext into an alternative form known as ciphertext so that the information remains private and is only readable by authorized parties. The ability to encrypt data is one of the primary security measures that companies use all over the world to ensure the safety and privacy of user information.

One of the earliest methods of encryption was invented by Julius Caesar during the rise of the Roman Empire in 60 BC and was used to encrypt messages to his correspondents, so that they could not be read if the message was intercepted by enemies. Augustus Caesar, his nephew, also used this method of encryption later in his life. Since then, encryption has come a long way, with modern encryption algorithms like *AES*, *RSA*, *DES*, etc... which are nearly impossible to break with modern technology (although these could be easily broken with quantum computers, but thats a topic for another day).

The Caesar Cipher

In this assignment, you are to implement a variation of the Caesar Cipher in C++ to enable encryption and decryption of messages. The cipher is described below.

Encryption

Encryption of data using the Caesar Cipher is accomplished by taking the original message (plaintext) and transforming each character using a **key**. The **key** is a number between 1 - 25 which indicates how many letters to *shift* when encrypting the plain text. For instance lets say a user selects the key value `3`, the key that Julius Caesar typically used with his cipher. We would then take the plaintext and shift each letter to the left by `3` letters.

For example each letter of the alphabet gets replaced with the following cipher text:

```
Plaintext:   ABCDEFGHIJKLMOPQRSTUVWXYZ
Ciphertext:  DEFGHIJKLMOPQRSTUVWXYZABC
```

where we can take a letter, `F` and shift it 3 letters to the right in the alphabet, which means that `F` becomes `I` in ciphertext.

In this program, we are going to take that one step further, by using letters of a passcode to describe how we rotate the characters (number of *shifts* to perform) in the plaintext message. For instance, providing the passcode `ABC` and the plaintext message `ATTACK AT DAWN` would become the following:

```
passcode: ABC
plaintext: ATTACK AT DAWN
ciphertext: AUVADM AU FAXP
```

where we start with the first letter of the message `A` and use the first letter of the passcode `A` to indicate how we *shift* the alphabet to get the correct character. Since the passcodes first letter is `A` and the first letter of the message is `A`, then the ciphertext character becomes `A` (*shifting* `A` by 0). The second letter of the message `T` is paired with the second letter of the passcode `B`. So the character becomes `T + 1` or `U` in the ciphertext. Then we move to the third letter of the message, using the third letter of the passcode to get `V` (i.e. `T + 2`). Then we repeat the process, starting over with the beginning of the passcode after every 3 characters. So the fourth character we would use `A` as the rotating key, and so on and so forth.

The more complex the passcode, the more secure your message will become!

Decryption

Decrypting a message involves the use of the same **passcode** that was used to encrypt the message. We simply take that passcode and use it to shift each letter of ciphertext `n` times to the left, where `n` is the rotation amount of each character of the passcode.

So for instance, given the encrypted message from above and where `passcode = ABC`, we can decrypt by starting with `A` and shifting that 0 letters to the left to become `A`, `U` and shifting that 1 letter to the left to become `T`, and so on and so forth, until we can re-create the message.

```
passcode: ABC
ciphertext: AUVADM AU FAXP
plaintext: ATTACK AT DAWN
```

Program Specifications

You are to create a program, that allows a user to encrypt or decrypt messages using the algorithm explained above. Your program must begin by displaying the following options to the user:

1. Encrypt Message
2. Decrypt Message
3. Quit

If a user selects options `1`, your program should prompt the user to enter a **passcode**, then use that passcode to encrypt a custom message entered via the keyboard.

If a user selects options `2`, you will first prompt the user to enter a **passcode** value, then prompt the user to enter the ciphertext to decrypt.

Your program should display the passcode, the plaintext and cipher text as the output of each option.

And finally if a user selects `3` you will exit the program.

Implementation Details

Your Program must follow the following guidelines and adhere to the following criteria

- Your program must re-prompt the user for a list of options after every encrypted or decrypted message is completed.
- Your program must account for incorrect options entered by a user (you may assume the user will always enter a valid number).
- Your program must contain at least 3 functions
    - You must implement a function using **pass by value**
    - You must implement a function using **pass by reference**
    - All functions must have a function prototype defined above `main()`, with the implementation defined below `main()`.

- You must validate the passcode according to the following:
  - Must be at least 1 letter
  - Must only contain alpha characters (a - z, A - Z).
  - If whitespace is included, you may ignore it (i.e. convert `My Password` to `MyPassword` to use in your encryption / decryption algorithm)
- If whitespace or punctuation is included in the message, you may ignore encrypting those characters. Only encrypt alpha characters (i.e. a - z and A - Z)
- The output of your encrypted message must be in ALL CAPS (as the examples below show).
- A whitespace or special character should not use up a character of the `passcode`. For instance given the passcode `SMU` and the message `"A test"`, you use `S` as the rotation for `A`, and `M` for the rotation of the next alpha character `T`, skipping the rotation of the whitespace character.

Hints

- Start early and start small! Concentrate on one feature at a time rather than trying to write the whole program in one sitting. A good way to break up the problem would be
  - Write a function that implements converting a single character to the correct ciphertext character given a rotation value and do the same thing for decrypting
  - Then use that function to encrypt / decript a whole string given a single passcode value
  - Then add the logic for a passcode where we use each character of the passcode to indicate the rotation or shift value.
  - Save the menu driven program part for last! Concentrate on the core operations of the program first!
- HINT: Use the `toupper()` function provided by `#include <cctype>` to convert any lower case letters to upper case before encrypting or decrypting
- HINT: Use `getline()` to read in a full line of text as a string!
- **Be sure to read the complete Programming Guidelines document at https://s2.smu.edu/~etchison/cs1342/pguide.doc for requirements on program structure and program comments.**

Example Output

```
Starting up your cipher!

Please select an option:
  1. Encrypt Message
  2. Decrypt Message
  3. Quit
Enter:
```

User enters 1

```
Enter your passcode: SMU
Enter the message to be encrypted: Attack at dawn

Encrypting your message...

Passcode: SMU
Plaintext Message: ATTACK AT DAWN
Ciphertext Message: SFNSOE SF XSIH

Please select an option:
  1. Encrypt Message
  2. Decrypt Message
  3. Quit
Enter:
```

User enters 2

```
Enter the message to be decrypted: SFNSOE SF XSIH
Enter your passcode: SMU

Decrypting your message...

Passcode: SMU
Ciphertext Message: SFNSOE SF XSIH
Plaintext Message: ATTACK AT DAWN

Please select an option:
  1. Encrypt Message
  2. Decrypt Message
  3. Quit
Enter:
```

User enters 3

```
Goodbye!
```

Additional Examples to test with

Feel free to test your programs with these additional examples below!

1. Plaintext Message: `The quick brown fox jumps over the lazy dog`
   Passcode: `PASSWORD`
   Ciphertext: `IHW IQWTN QRGOJ TFA YUEHO CMHG TZW HOQB SOY`

2. Plaintext Message: `The Buccaneers are the super bowl champs!`
   Passcode: `TOMBRADY`
   Ciphertext: `MVQ CLCFYGSQSJ AUC MVQ TLPHP UCIM THDKIG!`

3. Plaintext Message: `I'm gonna make him an offer he can't refuse.`
   Passcode: `GODFATHER`
   Ciphertext: `O'A JTNGH QRQS KNM TU SWLSU ME VHR'K XSIZSX.`

4. Plaintext Message: `Toto, I've a feeling we're not in Kansas anymore.`
   Passcode: `OZ`
   Ciphertext: `HNHN, W'US Z TDSKWMU VS'QS MCS WM YZBROR OMMLCQS.`

5. Plaintext Message: `World changers shaped here`
   Passcode: `SMU`
   Ciphertext: `OALDP WZMHYQLK EBSBYV TYJQ`

Submission Details

You must submit your program via GitHub (following the same instructions used for Lab02 and Program 1) by **6:00am CST, Monday February 22, 2021**. You may submit by **6:00am CST, Wednesday February 24, 2021** for a 30 point penalty.