BACKGROUND INFORMATION:

A cryptosystem is a system for encryption and decryption. Encryption, also called encoding or enciphering, involves changing the original message into a secret message, while decryption, also called decoding or deciphering, involves just the opposite – changing the secret message back into its original readable form. In cryptography, a *plaintext* file is a file containing data in its original, or readable, form while a *ciphertext* file is a file containing data in coded, or scrambled, form.

In this assignment, you will be creating a polyalphabetic cipher based on substitution using multiple substitution alphabets. In this cipher, instead of a single key, multiple one-letter keys are used to encrypt the original message by shifting the original letter to a new, encrypted letter corresponding to the key. For example, consider the following modified case from *Wikipedia* where the plaintext to be encrypted is: `ATTACK AT DAWN`. Then, the person sending the message chooses a keyword and repeats it until it matches the length of the plaintext so that the key `LEMON` becomes `LEMONL EM ONLE` (accounting for the spaces in the plaintext message which are not encrypted or decrypted). Then, the first letter of the plaintext, `A`, is paired with `L`, the first letter of the key. Assuming the shift starts with `A = 0`, `B = 1`, etc., we find that `L` is 11, so the plaintext `A` "shifts" 11 places in the alphabet to ciphertext `L`. The second letter of the plaintext, `T`, gets paired with `E`, the second letter of the key, so it shifts 4 (the shift value of `E`) places in the alphabet to ciphertext `X`. However, note that the third letter of the plaintext, `T`, gets mapped to a different ciphertext because It is paired with `M`, the third letter of the key. Of interest, is that this ciphertext wraps around to `F` based on shifting 12 places in the alphabet. The rest of the plaintext is enciphered in a similar fashion:

```
Plaintext:      ATTACK AT DAWN

Key:            LEMONL EM ONLE

Ciphertext:     LXFOPV EF RNHR
```

Note that since white space (or any unsupported character) is not encrypted or decrypted, the key should not be used (or incremented) so that the blank space does not take up a letter in the key. Mathematically, each letter of the plaintext and key can be given a number and a formula can be derived to encrypt for `c` (i.e., ciphertext) and `m` (i.e., plaintext) using `k` (i.e., key) as follows:

$$c = (m + k) \% 26$$

$$m = (26 + c - k) \% 26$$

Note that formula assumes `A` is 0, `B` is 1, and so forth with nothing to distinguish between uppercase `A` and lowercase `a`. Since you will most likely be using the character ASCII values, you will have to modify these formulas to fit your needs, but this should give you a place to start.

PROGRAM DESCRIPTION:

The purpose of this programming project is write a C++ program to implement a simple polyalphabetic cryptosystem that incorporates topics related to file I/O. In particular, this programming assignment will read an input file, encrypt or decrypt the file based on the requested operation, and write the results to an output file.

REQUIREMENTS:

- As with all programs in this course, your program's output should initially display the department and course number, your name, your EUID, and your e-mail address. This functionality will be implemented using a function.
- Your will prompt the user whether he/she would like to encrypt or decrypt a file. If a valid response is not input, you are to repeatedly re-prompt the user until a valid response is input.
- You will then prompt the user to enter the name of the input file to read from and the output file to write the resulting plaintext or ciphertext as appropriate. If there is a problem opening the file, you will display a meaningful error and exit the program using the appropriate exit status. You may assume that the length of the file name does not exceed 25 characters (26 with the terminating null character).
- If there are no errors causing you to terminate the program, you will then call a function that will process the input file (i.e., encrypt or decrypt the input file, as appropriate) and write the results to an output file based on the following requirements:
    1. You will prompt the user to enter a 5-letter key that will be used to encrypt or decrypt the file as appropriate for the operation. You must read this 5-letter key as 5 separate characters (i.e., you must five different `char` variables to account for each letter in the key). You may assume that the 5-letter key consists of alphabetic characters, but they may be entered as either uppercase or lowercase. Ultimately, you may treat each character as uppercase so that your shift will have a range of 0 – 25 (where `A` = 0, `B` = 1, etc.).
    2. The files that you will encrypt or decrypt will only contain uppercase alphabetic characters (`A` – `Z`), lowercase alphabetic characters (`a` – `z`), and whitespace, which may include any white space character such as a blank space, a tab, or a newline character.
    3. The user-defined function to process the input and output files should accept two parameters as a minimum, the input file stream and the output file stream, but it may utilize additional parameters as needed. You must process each file character-by-character (i.e., using the `get` and `put` member functions).

4. You will handle *encryption* in the following manner:

    a. You will encrypt all alphabetic characters, both uppercase and lowercase, using the correct user-entered key (either the first, second, third, fourth, or fifth letter, based on the order it falls within the plaintext file). Uppercase characters should encrypt to uppercase characters and lowercase characters should encrypt to lowercase characters. For example, if the key is `D`, `A` would be replaced with the `D`, `B` would be replaced by `E`, …, `Y` would be replaced by `B`, and `Z` would be replaced by `C`. Similarly, if the key is `D`, `a` would be replaced with the `d`, `b` would be replaced by `e`, …, `y` would be replaced by `b`, and `z` would be replaced by `c`.

    b. If the plaintext file contains white space, such as a blank space, a tab, or a newline character, you will do no encryption, but simply keep the white space character as is and write it to the ciphertext file.

    c. No other characters, such as punctuation or special characters, will be encrypted, although you may want to print out an error message and continue encrypting if it occurs, but do not write the non-valid character to the ciphertext.

    d. All encrypted characters based on these requirements will be written to the ciphertext file specified by the user.

5. You will handle *decryption* in the following manner:

    a. You will decrypt all alphabetic characters, both uppercase and lowercase, using the correct user-entered key (either the first, second, third, fourth, or fifth letter, based on the order it falls within the ciphertext file). Uppercase characters should decrypt to uppercase characters and lowercase characters should decrypt to lowercase characters. For example, if the key is `D`, `D` would be replaced with the `A`, `E` would be replaced by `B`, …, `B` would be replaced by `Y`, and `C` would be replaced by `Z`. Similarly, if the key is `D`, `d` would be replaced with the `a`, `e` would be replaced by `b`, …, `b` would be replaced by `y`, and `c` would be replaced by `z`.

    b. If the ciphertext file contains white space, such as a blank space, a tab, or a newline character, you will do no decryption, but simply keep the white space character as is and write it to the plaintext file.

    c. No other characters such as punctuation or special characters, will be decrypted, although you may want to print out an error message and continue decrypting if it occurs, but do not write the non-valid character to the plaintext.

d. All decrypted characters based on these requirements will be written to the plaintext file specified by the user.

- Be sure to close both the input and output file after you are done processing them.
- See the sample program runs for examples of what is expected. You should contact your instructor if there is any question about what is being asked for.
- Your code should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, course section, date, and brief description), comments for <u>each</u> variable, and commented blocks of code.
- Your program source code should be named "**`homework4.cpp`**", without the quotes.
- Your program will be graded based largely on whether it works correctly on the CSE machines (e.g., `cse01`, `cse02`, …, `cse06`), so you should make sure that your program compiles and runs on a CSE machine.
- This is an individual programming assignment that must be the sole work of the individual student.

You may assume that all input will be of the appropriate data type, although the value itself may not be valid.

You shall use techniques and concepts discussed in class – you are not to use global variables, `goto` statements, or other items specifically not recommended.

DESIGN (ALGORITHM):

On a piece of paper (or word processor), write down the algorithm, or sequence of steps, that you will use to solve the problem. You may think of this as a "recipe" for someone else to follow. Continue to refine your "recipe" until it is clear and deterministically solves the problem. Be sure to include the steps for prompting for input, performing calculations, and displaying output.

You should attempt to solve the problem by hand <u>first</u> (using a calculator as needed) to work out what the answer should be for a few sets of inputs. Calculations could include an actual formula for how the encryption/decryption is done and the expected results based on a key.

Type these steps and calculations into a document (i.e., Word, text, PDF) that will be submitted along with your source code. Note that if you do any work by hand, images (such as pictures) may be used, but they must be clear and easily readable. This document shall contain <u>both</u> the algorithm and any supporting hand-calculations you used in verifying your results.

SAMPLE OUTPUT (input shown in **bold green**):

```
mat0299@faculty:~/csce1030$ more plaintext1
ABCDEFGHIJKLMNOPQRSTUVWXYZ
mat0299@faculty:~/csce1030$ ./a.out





Would you like to ENCRYPT or DECRYPT a file (E or D)? h
Would you like to ENCRYPT or DECRYPT a file (E or D)? R
Would you like to ENCRYPT or DECRYPT a file (E or D)? E
Enter the name of your input file you want to encrypt: plaintext1
Enter the name of the output file to write the ciphertext: ciphertext1APPLE
Enter a 5-letter key to encrypt file: APPLE
mat0299@faculty:~/csce1030$ more ciphertext1APPLE
AQROIFVWTNKABYSPFGDXUKLICZ
mat0299@faculty:~/csce1030$ ./a.out
Would you like to ENCRYPT or DECRYPT a file (E or D)? e
Enter the name of your input file you want to encrypt: plaintext1
Enter the name of the output file to write the ciphertext: ciphertext1QWERT
Enter a 5-letter key to encrypt file: qwert
mat0299@faculty:~/csce1030$ more ciphertext1QWERT
QXGUXVCLZCAHQEHFMVJMKRAORP
mat0299@faculty:~/csce1030$ ./a.out
Would you like to ENCRYPT or DECRYPT a file (E or D)? D
Enter the name of your input file you want to decrypt: ciphertext1APPLE
Enter the name of the output file to write the plaintext: ptext1APPLE
Enter a 5-letter key to decrypt file: APPLE
mat0299@faculty:~/csce1030$ more ptext1APPLE
ABCDEFGHIJKLMNOPQRSTUVWXYZ
mat0299@faculty:~/csce1030$ ./a.out
Would you like to ENCRYPT or DECRYPT a file (E or D)? d
Enter the name of your input file you want to decrypt: ciphertext1QWERT
Enter the name of the output file to write the plaintext: ptext1QWERT
Enter a 5-letter key to decrypt file: QWERT
mat0299@faculty:~/csce1030$ more ptext1QWERT
ABCDEFGHIJKLMNOPQRSTUVWXYZ
mat0299@faculty:~/csce1030$ more plaintext2
Cowards die many times before their deaths The valiant never taste of death but once
William Shakespeare in Julius Caesar
mat0299@faculty:~/csce1030$ ./a.out
```

```
Would you like to ENCRYPT or DECRYPT a file (E or D)? e
Enter the name of your input file you want to encrypt: plaintext2
Enter the name of the output file to write the ciphertext: ciphertext2LEMON
Enter a 5-letter key to encrypt file: LEMON
mat0299@faculty:~/csce1030$ more ciphertext2LEMON
Nsioeow pwr xezm gtqqg opjafr elqwe oimhud Xts ilpuoae rqjrc xmggp sr rrlxt phe szqr
Hmxzvlq Evnviedrlvq wa Uyxwhd Gmsflv
mat0299@faculty:~/csce1030$ ./a.out

Would you like to ENCRYPT or DECRYPT a file (E or D)? d
Enter the name of your input file you want to decrypt: ciphertext2LEMON
Enter the name of the output file to write the plaintext: ptest2LEMON
Enter a 5-letter key to decrypt file: lemon
mat0299@faculty:~/csce1030$ more ptext2LEMON
Cowards die many times before their deaths The valiant never taste of death but once
William Shakespeare in Julius Caesar
mat0299@faculty:~/csce1030$ more plaintext3
While the loveRs and theoloGians have    Long made use of secret messages  the

overwhelming use      of cryptoGraphy has Historically been in diplomacy aND the
military
mat0299@faculty:~/csce1030$ ./a.out

Would you like to ENCRYPT or DECRYPT a file (E or D)? e
Enter the name of your input file you want to encrypt: plaintext3
Enter the name of the output file to write the ciphertext: ciphertext3HELLO
Enter a 5-letter key to encrypt file: hello
mat0299@faculty:~/csce1030$ more ciphertext3HELLO
Dltws alp wcciCd ouh essvpzRwhrd soci    Wzbn qlos bwp zt zincsa qpdghkpd  hoi

                      zq qycaecNvlavf lld Vpwezfpglwzf fppb pr otdssxlqf eYO hoi
zgsyaspztmyr izi
xtzpxlcm
mat0299@faculty:~/csce1030$ ./a.out
Would you like to ENCRYPT or DECRYPT a file (E or D)? D
Enter the name of your input file you want to decrypt: ciphertext3HELLO
Enter the name of the output file to write the plaintext: ptext3HELLO
Enter a 5-letter key to decrypt file: HELLO
mat0299@faculty:~/csce1030$ more ptext3HELLO
While the loveRs and theoloGians have    Long made use of secret messages  the

overwhelming use      of cryptoGraphy has Historically been in diplomacy aND the
military
```

```
mat0299@faculty:~/csce1030$ more plaintext4
plaintext4: No such file or directory
mat0299@faculty:~/csce1030$ ./a.out
Would you like to ENCRYPT or DECRYPT a file (E or D)? e
Enter the name of your input file you want to encrypt: plaintext4
ERROR: Unable to open file: plaintext4. Terminating...
```

TESTING:

Test your program to check that it operates as desired with a variety of inputs, especially boundary values or error conditions. Then, compare the answers your code gives with the ones you get from hand calculations.

Notice how, when using the same 5-letter key to encrypt and decrypt, an encrypted file will decrypt back to the original file. If you decrypt with a different key than what was used to encrypt, you will get a different result.

If you want to use any of the plaintext or ciphertext files above for your testing, simply copy and paste the text into an editor on a CSE machine.

SUBMISSION:

Your program will be graded based largely upon whether it works correctly on the CSE machines, so you should make sure your program compiles and runs on the CSE machines.

Your program will also be graded based upon your program style. This means that you should use comments (as directed), meaningful variable names, and a consistent indentation style as recommended in the textbook and in class.

- Program Header Example:

```
/*
 ============================================================================
 Name        : homework2.cpp
 Author      : Mark A. Thompson
 Version     :
 Copyright   : 2015
 Description : The program performs simple arithmetic operations based on in-
               put from the user.
 ============================================================================
 */
```

- Function Header Example:

```
/*
 ============================================================================
 Function    : deposit
 Parameters  : a double representing account balance and a double represent-
```

```
              ing the deposit amount
Return      : a double representing account balance after the deposit
Description : This function computes the account balance after a deposit.
=============================================================================
*/
```

We will be using an electronic homework submission on Blackboard to make sure that all students hand their programming projects on time. You will submit both (1) the program source code file and (2) the algorithm design document to the **Homework 4** dropbox on Blackboard by the due date and time.

Note that this project must be done individually. Program submissions will be checked using a code plagiarism tool against other solutions, so please ensure that all work submitted is your own.

Note that the dates on your electronic submission will be used to verify that you met the due date and time above. All homework up to 24 hours late will receive a 50% grade penalty. Later submissions will receive zero credit, so hand in your best effort on the due date.

As a safety precaution, do not edit your program (using `vi` or `pico`) after you have submitted your program where you might accidentally re-save the program, causing the timestamp on your file to be later than the due date. If you want to look (or work on it) after submitting, make a copy of your submission and work off of that copy. Should there be any issues with your submission, this timestamp on your code on the CSE machines will be used to validate when the program was completed.