CSCI 689 **Computer Assignment 3** **Fall 2017**

(10 points)

Caesar Cipher Updated

The encryption technique called **Caesar cipher** replaces each letter in a text message by a letter that appears a fixed distance in the alphabet. As an example, suppose that you wanted to encode a message by shifting every letter ahead three places. In this cipher, each A becomes a D, B becomes E, and so on. If you reach the end of the alphabet, the process cycles around to the beginning, so that X becomes A, Y becomes B, and Z becomes C.

Note that the transformation applies only to letters; any other characters are copied unchanged to the output, where the case of letters is unaffected: lowercase letters come out as lowercase, and uppercase letters come out as uppercase. You should also write your C++ program so that a negative value of shift means that letters are shifted toward the beginning of the alphabet instead of toward the end.

In addition to regular shifting, a second shifting is applied to each letter in the input stream, where a 26-letter key is used to figure out the shifting value of the letter. The key is entered from the stdin and it contains all 26 uppercase letters but a letter in the key could be in any of the 26 possible positions. As an example, the key could be const string key = "QWERTYUIOPASDFGHJKLZXCVBNM". For the letter A, the corresponding letter in the key is Q, so the second shift value for A is –16 that is the difference between the ASCII values of A and Q; for the letter B, the corresponding letter in the key is W that yields the second shift value of –21; for the letter C, the corresponding letter in the key is E that yields the second shift value of –2; etc. For a lowercase letter, you need to convert the letter to uppercase before you use.

There are two data files for this program: prog3.d1 and prog3.d2. Both are in directory: /home/cs689/progs/17f/p3. The first file contains several test values for shift and key, and the second one contains a text message to encode.

In addition to your C++ source file, you also need to have a header file, where you put declarations of all constants (number of letters in the alphabet for the text message and the full path name of the second data file) and function prototypes that you use in your program.

The main ( ) routine calls the following function for each shift and key values, entered from the stdin.

- void process_infile (const int& shift, const string& key): This opens the second data file, and if it is unsuccessful, it displaces an error message on stderr and exits the program with the exit value EXIT_FAILURE. It prints out the shift and key values on stdout passed as arguments and gets the text input from the data file. To process each input line in the data file, it calls the encodeCaesarCipher ( ) function, which is described below, and it prints out the encrypted text returned by this function on stdout. Finally, it closes the data file.

To implement a Caesar cipher, you should define the following function.

- string encodeCaesarCipher (string str, const int& shift, const string& key): It returns a new string formed by shifting every letter in str forward the number of letters indicated by shift and key, cycling back to the beginning of the alphabet if necessary. To implement shifting, this function calls the following auxiliary function.

- int new_position (const char& c, const int& shift, const string& key): For the character c, for the shift value, and for the second shift value obtained from the key, it returns the new position of c after shifting.

At the top of your source file, include its header file. To compile your source file and link its object file with the system library routines, execute: Make N=3 (assuming you name your source and header files as prog3.cc and prog3.h).

For a final test of your program, execute: Make execute N=3. When the execution is over, the output file prog3.out will contain the encrypted text for each shift and the key in file prog3.d1, as well as any error messages that might be generated by your program. See the correct output in file prog3.out in directory: ~cs689/progs/17f/p3. After you are done, delete the object and executable files of your program by executing: Make clean.

Submit your source and header files of your program to your TA by executing: mail_prog.689 prog3.cc prog3.h.