HW #01
CSCI-400 Spring 2013

The purpose of this assignment is just to familiarize yourself with some of the less commonly used (for many programmers) aspects of the C programming language that will be useful later in this course.

Your program will read the text from the input file and either encrypt or decrypt it, as determined by the mode, using the cipher key provided. You will use a Vigenere cipher, which is described later in this document. Remember that the goal of the assignment has nothing to do with cryptography, this is just a vehicle for exploring some C language aspects. Thus, some of the instructions for this program may seem a bit forced, largely because they are.

Your program will take up to four command line arguments. The first is the name of an input text file, the second is the name of the output text file, the third is the cipher key and the fourth is the mode. If the mode is omitted, then it is to be assumed that the mode is to encrypt the file. If the cipher key is omitted, then a default key of "Colorado School of Mines" is to be used. If the output file is omitted then the output is to be directed to the screen. The user is required to supply an input file – if they don't, issue an appropriate error message and terminate the program.

Output, to the screen, the command, with arguments, that was used to launch the program.

Output, to the screen, all four parameters that will be used.

The encryption/decryption process will only be concerned with the letters of the alphabet and will be case-insensitive. The input text (to be encrypted, known as the plaintext) can be any ASCII text file, meaning it may have upper/lower case letters, numbers, whitespace, punctuation, etc. Your program will ignore everything except upper/lower case letters. It will encrypt them (described later) and output the encrypted text (the ciphertext) in groups of five uppercase characters. When decrypting, the ciphertext should be groups of five letter characters, but your program will accept anything (just like the plaintext) and only consider the letters it finds (be they uppercase or lowercase).

Once the information is processed (i.e., encrypted or decrypted), your program will output some statistics about the data. Regarding the input file, it will output the total number of characters in the file as well as the total number of lines. It will then break this down as the number of total number of letters, number of uppercase, lowercase, digits, whitespace, and all else (punctuation and control codes). You will do the same for the output file. You may format this however you like, but it should be neat and easy to understand.

Your program is to read and process the input file one character at a time, including producing whatever output is appropriate. Your program may not use scanf() or any of its variants; instead, use fgetc() and similar functions to get input. Your program may not use any of the string or ctype library functions.

You are to write a function called LoadKey() that will accept a pointer to the string containing the key. This function will return a pointer to a block of dynamically allocated memory that holds a sanitized copy of the key. The key is sanitized by removing everything except letters and converting all letters to uppercase. If the user did not supply a key on the command line, then a NULL pointer is to be passed to the function and the function is to use the default key (which is to appear in your code listing as shown above, namely a mix of upper/lower case letters with spaces).

The actual encrypting and decrypting uses the relation C = P + K (mod 26), where C is the ciphertext character, P is the plaintext character, and K is the key character. Each letter, A-Z, has a value of 0-25 respectively. The characters are first converted to their values, the above relation is used to solve for the desired variable, and that value then converted back to a character. So, for instance, when encrypting 'Y' from the plaintext with the 'D' from the key, these are first converted to 24 and 3 respectively. Adding these yields 27 which is then reduced modulo 26 to yield a final value of 1. Converting this to a character results in 'B'. When decrypting this, the 'B' and 'D' would be converted to 1 and 3, they would then be subtracted (since now we are using P = C – K) to get -2 which, when reduced modulo 26, yields 24, thus eventually resulting in the character 'Y'.

The first character of the plaintext/ciphertext is encrypted/decrypted with the first character of the key. The second character of the input uses the second character of the key, and so on. If the end of the key is reached, the process continues by going back to the first character of the key.

Your program is expected to explicitly free memory and close files as appropriate and not to simply let these happen automatically when the program closes.

Your source code file should be named CS400_USERID_HW01.c where 'USERID' is replaced with your CSM Mines E-Mail Username (which should be the same as your ADIT Username). Then zip this file (even though it is a single file – later homeworks will involve more than one file) using the same file name, except with a .zip extension. You may include a Readme.txt file if you wish. If you chose to use user-defined header files or additional source code files, please name them in a similar fashion and include them in your zip file.

RUBRIC – 40 pts

10 - Command line parameters are processed properly
10 – LoadKey() function works per specification
5 – NULL pointer checks are made where appropriate
5 – Dynamic memory is allocated and freed properly
5 – Files are opened and closed properly
3 – Statistics are done properly
2 – Encryption/Decryption are done properly