

## **COMP 1130 Principles of Programming II**

### **Program 2 – Text Encryption**

**Maximum Possible Points: 20**

#### **Due Date:**

- All required files need to be submitted by September 27th

#### **Objectives:**

- To gain experience with file I/O.
- To gain experience with arrays.

#### **Overview:**

Encryption is the process of encoding a message so that its meaning is not obvious. Though this is an ancient art, it has become very important in the modern age. Encryption can be used to protect the confidentiality and integrity of electronic funds transfers, provide security for Internet transactions, and many other purposes. There are various encryption algorithms to convert a plain text into an unintelligible encoded text called a cipher text.

Write a C++ program to generate the cipher text for a given plain text file using a few rudimentary encryption algorithms. The program should read the plain text from an input file called plain.txt. Your program should generate two cipher texts (using two different techniques described next) for the input text and write those cipher texts into two output files called cipher1.txt and cipher2.txt. The input and output files can be found in Blackboard. Your program will produce the encrypted text converting one letter at a time. You may assume that all the characters are uppercase English letters without any punctuation. Your program must have four functions in addition to the main function. Each function should accept an array of characters for the plain text. The main function will be responsible only for creating the initial array for storing the plain text and calling the functions to write the encrypted text.

The **readFile** function will read the plain text file from disk and store into a character array. It should receive a single reference parameter from the main function, which is a character array in which to store the text read from the file.

The **getCaesarCipher** function will encrypt the text using a simple Caesar Cipher, which was used by Julius Caesar to communicate with his army. This algorithm shifts each letter in the message a fixed number of places down the alphabet. For example, using the 5-shift Caesar Cipher, the message, "GOOD BYE" would be encrypted as, "LTTI GDJ". In this example, 'G' is shifted to 'L', 'B' is shifted to 'G', 'Y' is shifted to 'D', and so on. You should notice that the shift is circular. In this program you will be implementing a 13-shift Caesar Cipher, also known as ROT

13 (rotate by 13 places).

The **getAtbashCipher** function will encrypt the text using the reversed alphabet where the first letter is replaced by the last one, the second letter is replaced by the second last one, and so on. For example, "GOOD BYE" would be encrypted as, "TLLW YBV".

The **showArray** function will output the current contents of a single character array parameter to the console.

**Instructions:**

- This is an individual programming assignment.
- Write your code in a file called prog2.cpp.
- Use meaningful variable names, helpful comments, and a consistent coding style.
- Your program file should have the appropriate comment block at the top:

```
/* Name: Your Name  
 * File Name: prog2.cpp  
 * Date: Day Month, Year  
 */ Program Description: brief description of the program
```

- All functions should have a header comment block describing the purpose of the function, input, and output:

```
*****  
 * Name: Name of the function.  
 * Description: Description of why the function exists and  
 * how it accomplishes its goal.  
 * Input: Parameter list with descriptions  
 * Output: Return value with description  
*****/
```

**Deliverables:**

You will need to submit the following in Blackboard by 2:59 PM on September 27th:

- The source file prog2.cpp and screenshots of your program execution.
- A report providing:
  1. The project specification,
  2. A brief summary stating, in a technical manner, how you arrived at your solution and what problems you encountered, if any, and
  3. Rigorous testing documentation (recreation of the sample executions + 1 more execution).

**Grading:**

This project is worth 30 points distributed as follows:

- Demo of Part I (10 pts)
  - Declaration and initialization of the character array (3 pts)
  - Successful implementation of readFile (3 pts)
  - Successful tests of showArray (3 pts)
- Part II (15 pts)
  - Successful implementation of main (5 pts)
  - Successful implementation of getCaesarCipher (5 pts)
  - Successful implementation of getAtbashCipher (5 pts)
- Program Style (3 pts)
  - Meaningful variable names (1 pt)
  - Proper indentation (1 pt)
  - Sufficient comments (1 pts)
- Report (2 pts)
  - Project specification (1 pt)
  - Solution summary (1 pt)

**YOU WILL LOSE 50% OF THE POINTS YOU RECEIVE IF YOUR PROGRAM DOES NOT COMPILE AND YOU DO NOT SUBMIT TWO SCREENSHOTS.**