**CSC 1101 – Problem Solving and Programming Laboratory**

**Lab 15 – Omar Faruk**

**25 points – Due November 9, 11pm**

**a)** Save this document with your name and the homework number somewhere in the file name.

**b)** Paste your code and screenshots into the document.

**c)** Submit this document and your .cpp file(s) to the Canvas item where you downloaded this document. Do not submit a zip file but individually attach your files.

You've been hired by *Binary Buzzards* to write a C++ console application that encodes and decodes text. Just like Lab 14, the text comes from two files: SampleText1.txt and SampleText2.txt. Encoding keys are stored in file RandomNumbers.txt. Place the files where your app can find them. File SampleText1 will be used to test your app. File SampleText2 will be used for your final run. File RandomNumbers.txt will provide numbers to shift character codes.

In your app, attempt to open the text file and random numbers file. Print an error message if the text file doesn't open. Otherwise, print an error message if the random numbers file doesn't open. Otherwise, read one line at a time from the text file and keep count of the number of text file lines read. For each text file line read, read one shift number from the random numbers file and keep count of the number of random numbers file lines read. Echo the line. Encode and print the line by calling function **encodedLine** (described below). Add the shift number used in parentheses after the line. Then decode and print the line by calling function **decodedLine** (described below). Add the shift number used in parentheses after the line. Close the two files and report the number of lines read from each.

**string encodedLine(string inLine, int shift)**

This value function takes the original line and shift as input and returns the equivalent encoded string. Start with an empty encoded string. Use a for loop to encode each character of the line. Use the following algorithm to encode a character and append the encoded character to the encoded string:

● Convert the character to an integer (cast to int).

● *Add* the shift to the integer value.

● Mod the integer value by 256 to insure its stays within the character range 0-255.

● Convert the integer back to a character (cast back to char).

This function does no printing, it just returns a string to the function that called it.

**string decodedLine(string inLine, int shift)**

This value function takes the encoded line and shift as input and returns the original string. It does the reverse of function **encodedLine**. The original string and decoded string should be identical. Use the following algorithm to decode a character and append the decoded character to the decoded string:

● Convert the character to an integer (cast to int).

● *Subtract* the shift from the integer value.

● Mod the integer value by 256 to insure its stays within the character range 0-255.

● Convert the integer back to a character (cast back to char).

This function does no printing, it just returns a string to the function that called it.

Declare global constants for the file names, column width, and character boundary (256). For file SampleText1, the output should look like this:

Welcome to Binary Buzzards

--------------------------

Reading lines from file 'SampleText1.txt' ...

Reading numbers from file 'RandomNumbers.txt' ...

Original: It has often been said

Encoded: É╗g»¿║g╢¡╗¼╡g⌐¼¼╡g║¿░½g (+71)

Decoded: It has often been said (-71)

Original: there's so much to be read

Encoded: ┐│░╜░r╛k╛║k╕└«│k┐║k¡░k╜░¼»k (+75)

Decoded: there's so much to be read (-75)

Original: you never can cram

Encoded: ╩└╞q┐╢╟╢├q┤▓┐q┤├▓╛q (+81)

Decoded: you never can cram (-81)

Original: all those words in your head.

Encoded: ÖññX¼áº½¥X»º¬£½XíªX▒º¡¬Xá¥Ö£f (+56)

Decoded: all those words in your head. (-56)

Original: So the writer who breeds

Encoded: Ç£MíòÆMñƒûíÆƒMñò£MÅƒÆÆæáM (+45)

Decoded: So the writer who breeds (-45)

Original: more words than he needs

Encoded: ║╝┐▓m─╝┐▒└m┴╡«╗m╡▓m╗▓▓▒└m (+77)

Decoded: more words than he needs (-77)

Original: is making a chore

Encoded: æ¢HòëôæûÅHëHïÉùÜìH (+40)

Decoded: is making a chore (-40)

Original: for the reader who reads.

Encoded: ò₧íOúùöOíöÉôöíOªù₧OíöÉôó] (+47)

Decoded: for the reader who reads. (-47)

Original: That's why my belief is

Encoded: råÆEæ>òåù>ïù>Çâèçâä>çæ> (+30)

Decoded: That's why my belief is (-30)

Original: the briefer the brief is

Encoded: òëåAâôèåçåôAòëåAâôèåçAèöA (+33)

Decoded: the briefer the brief is (-33)

Original: the greater the sigh

Encoded: ▓ªú^Ñ░úƒ▓ú░^▓ªú^▒ºÑª^ (+62)

Decoded: the greater the sigh (-62)

Original: of the reader's relief is.

Encoded: Ñ£V¬₧¢V¿¢ùÜ¢¿]⌐V¿¢óƒ¢£Vƒ⌐d (+54)

Decoded: of the reader's relief is. (-54)

12 line(s) read from file 'SampleText1.txt'.

12 numbers(s) read from file 'RandomNumbers.txt'.

End of Binary Buzzards

Use file SampleText2 for the final run pasted below.

*[your program code here]\**

//==========================================================

//

// Title: Encryption Decryption Text File

// Course: CSC 1101

// Lab Number: Lab 15

// Author: Omar Faruk

// Date: 11/08/2020

// Description:

// Taking two input files to encode and decode

// a set of string from the text file using functions

// and loops. Used file handling to open and close

// and complete the lab.

//==========================================================

#include <cstdlib> // For several general-purpose functions

#include <fstream> // For file handling

#include <iomanip> // For formatted output

#include <iostream> // For cin, cout, and system

#include <string> // For string data type

using namespace std; // So "std::cout" may be abbreviated to "cout"

// Globals

const int character\_boundary = 256;

const int COLMFT1 = 15, COLMFT2 = 40;

string encodedLine(string inLine, int shift)

{

int length = inLine.length();

int k;

char p;

string encode;

for (int i = 0; i < length; i++)

{

k = int(inLine[i]);

k = k + shift;

k = k % character\_boundary;

p = char(k);

encode = encode + p;

}

return encode;

}

string decodedLine(string inLine, int shift)

{

string decode;

int k, length = inLine.length();

char d;

for (int i = 0; i < length; i++)

{

k = int(inLine[i]);

k = k - shift;

k = k % character\_boundary;

d = char(k);

decode = decode + d;

}

return decode;

}

int main()

{

// Declare variables

ifstream SampleText\_inFile, RandomNumbers\_inFile;

int sample\_linecount = 0, numbers\_linecount = 0;

string sample\_current;

int numbers\_current;

// Declclare constants

const string SAMPLE\_TEXT\_INPUT = "SampleText2.txt", RANDOM\_NUMBERS\_INPUT = "RandomNumbers.txt";

// Show application header

cout << "Welcome to Binary Buzzards!" << endl;

cout << "---------------------------" << endl << endl;

// Open Input Files of Random Numbers and Sample Text

SampleText\_inFile.open(SAMPLE\_TEXT\_INPUT);

RandomNumbers\_inFile.open(RANDOM\_NUMBERS\_INPUT);

if (!SampleText\_inFile.is\_open())

cout << "Error: unable to open file '" << SAMPLE\_TEXT\_INPUT << "'." << endl << endl;

if (!RandomNumbers\_inFile.is\_open())

cout << "Error: unable to open file '" << RANDOM\_NUMBERS\_INPUT << "'." << endl << endl;

else

{

// Reading lines

cout << "Reading lines from '" << SAMPLE\_TEXT\_INPUT << "' ..." << endl

<< "Reading lines from '" << RANDOM\_NUMBERS\_INPUT << "' ..." << endl << endl;

// Loop to read lines from files

while (SampleText\_inFile.good() && RandomNumbers\_inFile.good())

{

// Reading sample text and printing original output

getline(SampleText\_inFile, sample\_current);

cout << setw(COLMFT1) << left << "Original:" <<

setw(COLMFT2) << left << sample\_current << endl;

sample\_linecount = sample\_linecount + 1;

// Reading Random numbers

RandomNumbers\_inFile >> numbers\_current;

numbers\_linecount = numbers\_linecount + 1;

// Printing out encoded and decoded

cout << setw(COLMFT1) << left << "Encoded:" <<

setw(COLMFT2) << left << encodedLine(sample\_current,numbers\_current) <<

"(+"<< numbers\_current << ")" << endl;

cout << setw(COLMFT1) << left << "Decoded:" <<

setw(COLMFT2) << left << decodedLine(encodedLine(sample\_current, numbers\_current), numbers\_current)

<<"(-" << numbers\_current << ")" << endl << endl;

}

// Close input files

SampleText\_inFile.close();

RandomNumbers\_inFile.close();

// Line count of both input files

cout << sample\_linecount << " line(s) read from file '" << SAMPLE\_TEXT\_INPUT << "'" << endl

<< numbers\_linecount << " number(s) read from file '" << RANDOM\_NUMBERS\_INPUT << "'" << endl;

}

// Show application close

cout << "\nEnd of Binary Buzzards" << endl;

}

*[your program output here]\*\**



