ECE4574 FA23 - Prof. Jones – HW 1

Due Friday, September 22, 2023 – 11:59 PM via Canvas

You are to write and test a C++ program that will demonstrate simple text encoding and decoding. The idea behind text encoding is to accept a string of printable text characters, transform them in some (usually reversible) way, and write the output. I want your text encoders to allow for both encoding and decoding, using a common interface described below.

Functional Requirements

1. The system shall ask the user for a text string, accept the input string as a QString and write out to the console the string accepted.
2. The system shall accept input strings up to 1000 characters long.
3. All strings written out shall be written to the console.
4. All strings written out shall be written out as follows. First, write an (optional) prefix label; refer to my example output. Next, write the total length of the string in decimal, surrounded by angle brackets, then write the string surrounded by square brackets. So, if the string "`Hello, world!`" is the input, you should write out: **`Input: <13>[Hello, world!]`**
5. The system shall implement "flip" encoding.
    a. "flip" encoding operates on the ASCII (or Unicode) character codes. The printable characters have ASCII codes between 0x20 and 0x7E inclusive. Character 0x20 should become 0x7E, 0x21 should become 0x7D, …, character 0x7E should become 0x20. Nonprintable characters should be unchanged.
6. The system shall implement "flip" decoding, which is the reverse of the flip encoding.
7. The system shall implement "one-time pad" or OTP encoding.
    a. OTP encoding also operates on the ASCII (or Unicode) character codes. When the OTPEncoder object is created, it should create a short array of integer offsets (I used **`int pad[] = {14, 32, 6, 8, 0, 24, 43, 8, 62, 7, 29}`**. You should use a different pad; <u>the pad length should be a prime number</u>. Each character in the string is encoded by adding the corresponding offset to the character's ASCII code; the first character in my example has 14 added to it, the second has 32 added to it, etc. When you reach the end of the pad, go back to the beginning. If the addition results in a value greater than 0x7E, subtract the proper constant to bring it back into the range 0x20 to 0x7E – so 0x70 + 15 = 0x7F should become 0x20, etc. Nonprintable characters should be unchanged by this encoding.
8. The system shall implement OTP decoding, which is the reverse of OTP encoding.
9. The system shall implement Invert Case coding, in which all upper-case letters are converted to lower-case and all lower-case letters are converted to upper-case (non-letters are unchanged).
10. The system shall implement Invert Case decoding, which is the reverse of Invert Case coding.
11. The system shall implement a demo and test mode, as follows. When the program starts, it should do the following:
    a. Ask the user for an input string, and write it out.

b. Apply flip encoding to the input, and write out the result.
c. Apply flip decoding to the flipped input and write out the result.
d. Apply OTP coding to the original input and write out the result.
e. Apply OTP decoding to the OTPed input and write out the result.
f. Apply Invert Case coding to the original input and write out the result.
g. Apply Invert Case decoding to the inverted input and write out the result.
h. An example of this set of operations can be seen in my console output, below (I have inserted extra line feeds for clarity).

Non-functional Requirements

1. This is to be a Qt console application – no windows or widgets; all IO happens through the console window.
2. Each text encoder should be implemented in a separate C++ class, embodied in two files: a .h for the class declaration and a .cpp for the class implementation.
3. You will be creating three classes: FlipCoder, OTPCoder and InvertCaseCoder. Each should be in its own pair of source files (a .h and a .cpp). For each encoder, the interface should be exactly the same; for example:

```
OTPCoder();           // a constructor
QString encode(const QString);
                      // accepts a QString and returns result in a QString
QString decode(const QString);
                      // accepts a QString and returns result in a QString
```

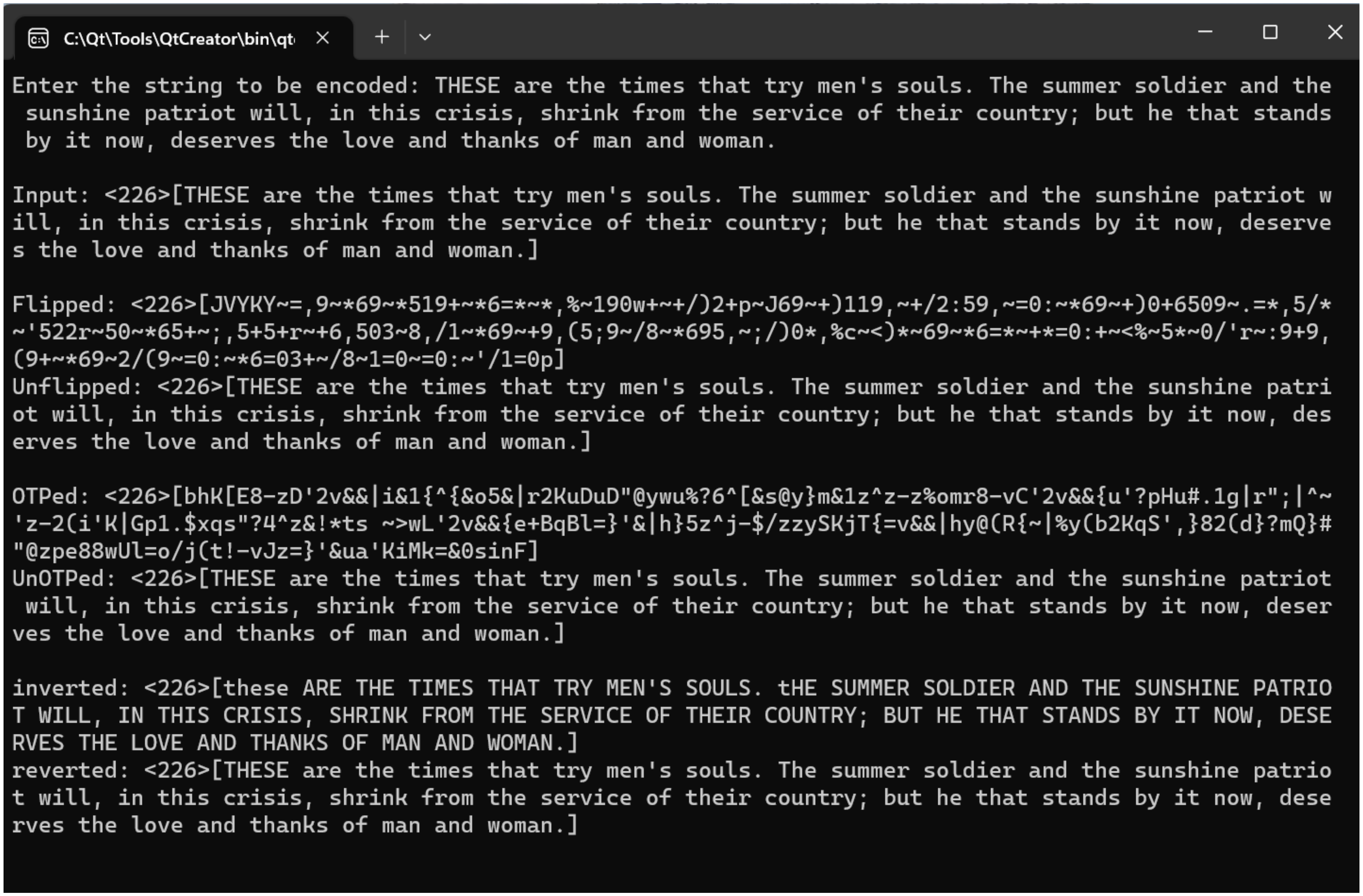
4. All files should have a file header, similar to the one shown below.
5. Add comments to your code when they would help the reader understand what's going on.
6. Variable names should somehow reflect the meaning of the quantity (no "var1" or "temp").

SUBMISSION

Your submission will consist of a Word or pdf file and a .zip file containing all of your source code. Do NOT put the Word file in the zip file! Your Word file should contain:

- your complete C++ code, both .h and .cpp files (pasted in as plain text, no screenshots or dark mode);
- the console output of your program operating on three test strings, pasted in as plain text; use the following:
  - your full name, including spaces
  - "**`In a hole in the ground there lived a hobbit. 1234567890~!@#$%^&*()`**"
  - another test string of your choice, of at least 20 characters.

Here is the output from my program on a sample text string:

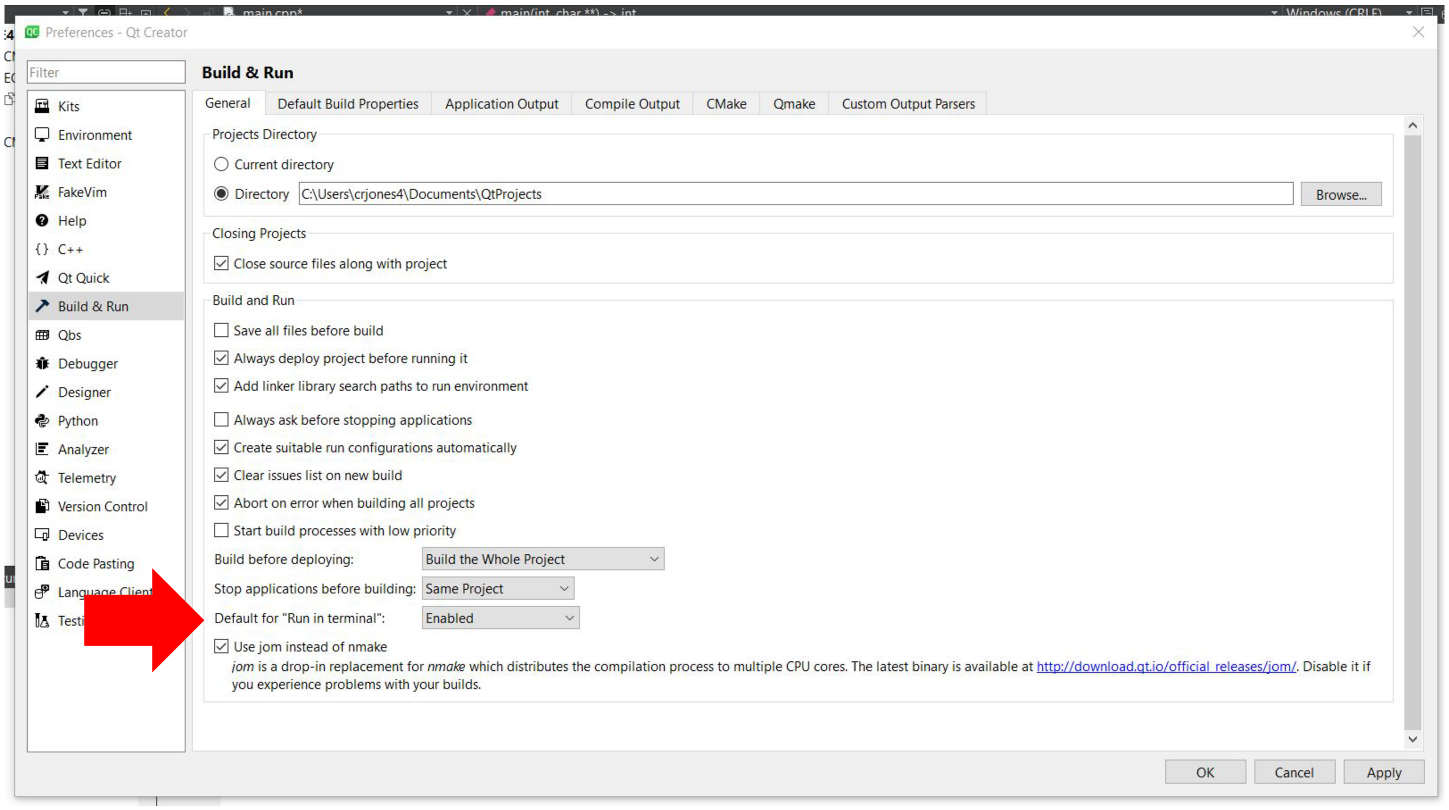


C:\Qt\Tools\QtCreator\bin\qt

```
Enter the string to be encoded: THESE are the times that try men's souls. The summer soldier and the
 sunshine patriot will, in this crisis, shrink from the service of their country; but he that stands
 by it now, deserves the love and thanks of man and woman.

Input: <226>[THESE are the times that try men's souls. The summer soldier and the sunshine patriot w
ill, in this crisis, shrink from the service of their country; but he that stands by it now, deserve
s the love and thanks of man and woman.]

Flipped: <226>[JVYKY~=,9~*69~*519+~*6=*~*,%~190w+~+/)2+p~J69~+)119,~+/2:59,~=0:~*69~+)0+6509~.=*,5/*
~'522r~50~*65+~;,5+5+r~+6,503~8,/1~*69~+9,(5;9~/8~*695,~;/)0*,%c~<)*~69~*6=*~+*=0:+~<%~5*~0/'r~:9+9,
(9+~*69~2/(9~=0:~*6=03+~/8~1=0~=0:~'/1=0p]
Unflipped: <226>[THESE are the times that try men's souls. The summer soldier and the sunshine patri
ot will, in this crisis, shrink from the service of their country; but he that stands by it now, des
erves the love and thanks of man and woman.]

OTPed: <226>[bhK[E8-zD'2v&&|i&1{^{&o5&|r2KuDuD"@ywu%?6^[&s@y}m&1z^z-z%omr8-vC'2v&&{u'?pHu#.1g|r";|^~
'z-2(i'K|Gp1.$xqs"?4^z&!*ts ~>wL'2v&&{e+BqBl=}'&|h}5z^j-$/zzySKjT{=v&&|hy@(R{~|%y(b2KqS',}82(d}?mQ}#
"@zpe88wUl=o/j(t!-vJz=}'&ua'KiMk=&0sinF]
UnOTPed: <226>[THESE are the times that try men's souls. The summer soldier and the sunshine patriot
 will, in this crisis, shrink from the service of their country; but he that stands by it now, deser
ves the love and thanks of man and woman.]

inverted: <226>[these ARE THE TIMES THAT TRY MEN'S SOULS. tHE SUMMER SOLDIER AND THE SUNSHINE PATRIO
T WILL, IN THIS CRISIS, SHRINK FROM THE SERVICE OF THEIR COUNTRY; BUT HE THAT STANDS BY IT NOW, DESE
RVES THE LOVE AND THANKS OF MAN AND WOMAN.]
reverted: <226>[THESE are the times that try men's souls. The summer soldier and the sunshine patrio
t will, in this crisis, shrink from the service of their country; but he that stands by it now, dese
rves the love and thanks of man and woman.]
```

CONSOLE APP

Your program should be a console app created in Qt. When you create a new project, choose the “Qt Console Application” project type. Note that you may not (probably won’t) see the console window when you run your program! To enable the console window to appear, you may have to change a project setting in Qt. To do this, in the menu system choose Edit…Preferences and select the Build & Run page; on the General tab, the “Default for Run in terminal’” setting must be ON, as shown here:



Build & Run

Projects Directory

Current directory [✓ Directory] C:\Users\crjones4\Documents\QtProjects

Closing Projects

[✓] Close source files along with project

Build and Run

[ ] Save all files before build
[✓] Always deploy project before running it
[✓] Add linker library search paths to run environment
[ ] Always ask before stopping applications
[✓] Create suitable run configurations automatically
[✓] Clear issues list on new build
[✓] Abort on error when building all projects
[ ] Start build processes with low priority

Build before deploying: Build the Whole Project
Stop applications before building: Same Project
Default for "Run in terminal": Enabled

[✓] Use jom instead of nmake
jom is a drop-in replacement for *nmake* which distributes the compilation process to multiple CPU cores. The latest binary is available at http://download.qt.io/official_releases/jom/. Disable it if you experience problems with your builds.

Here is a short version of my main.cpp file (this only uses one encoder):

```
// main.cpp     Creed Jones     VT      August 21, 2023
// This is the top level routine for ECE4574 FA23 HW1
// Simple console app

#include <QCoreApplication>

#include "EncoderUtils.h"               // I chose to put the writeString() function in a separate file
                                        // my function writes a string in the format I requested
                                        // you don't have to do it in a separate file - but it's good practice

void HW1process();

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    HW1process();

    return a.exec();
}

void HW1process()            // The function that executes the whole HW1 process
{
    FlipCoder flipper;

    QString inputstr;
    QTextStream strmin(stdin);
// 1.   Ask the user for a text string (see below) and write out to the console the string accepted.
    printf("Enter the string to be encoded: ");
    inputstr = strmin.readLine();
    writeString(inputstr, "Input:");

// 2.   Perform "flip" encoding and write the result to the console.
    QString flipped = flipper.encode(inputstr);
    writeString(flipped, "Flipped:");

// 5.   Perform "flip" decoding, which is the reverse of the flip encoding, and write the result to the console.
    QString unflipped = flipper.decode(flipped);
    writeString(unflipped, "Unflipped:");
    printf("\n");

}
```