CSE 1384 - Recursion
Lab 10

Objectives:

- Continue practicing past concepts
- Practice building recursive functions in C++

Background:

This week's lab will be involving Caesar ciphers. As such, here's some background/explanation as to what that specifically is. Essentially, a Caesar cipher takes a word or phrase and shifts the letters of the text by a certain amount. So, a shift 3 would look something like this:

```
Maggie Mae Neal → Pdjjlh Pdh Qhdo
```

Where this is the alphabet associated with it:

Normal alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Encoded alphabet	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

(Note: You can shift right or left – we are shifting right for encoding in this lab.)

Notice how the alphabet wraps around, where Z is no longer the end and A is no longer the beginning. Also note, to decode, you'll shift opposite what you did to encode.

Please find further sources on Caesar ciphers here:

- Caesar Cipher Decoder
- Caesar Cipher wikipedia

Assignment Prep:

To help you guys out a simple Caesar cipher program has been included (`sample-cipher.cpp`). It shifts to the left and by 8 – it also only encodes. So, not exactly the shift we're doing, but it provides a nice jumping off point. Here's a sample of it running:

```
Here's a sample cipher! We're shifting left by 8 to encode in this sample.

Enter in a word/phase to encode: This is a sample.

Starting point: This is a sample.
Ending point: Lzak ak s ksehdw.
```

Assignment:

Your job is to build a Cipher recursive function that works inside the file (`main.cpp`) provided for you. It *must* work with my main function and you cannot change anything about it. The main file will use the function you complete and provide a looping menu to encode and decode messages.

Your recursive function is already defined – you cannot alter the parameters or how the menu calls the function. The function is defined like so:

```
string caesarShift(string phrase, int position, bool encode);
```

Phrase → the word you're either encoding or decoding
Position → the number index you currently are at in the word (you start at 0 for the first call)
Encode → boolean true/false. True for encoding, false for decoding

You will be CHANGING the phrase you take into the function in order to encode or decode it. This is normal and expected.

If the boolean denotes encoding, you should be encoding (shift right), otherwise it's decoding and you'll need to shift left.

Please take note of the milestones because the hints of how I'd approach it are built-in to how the milestones are ordered and built.

***** NOTICE: A shift 5 is what you should be using for this lab on default. *****

Milestones:

You must receive a check-off for each item to receive full credit on the lab. You may show the milestones incrementally, or receive a check off for multiple in one demo depending on your progress.

Examples of each milestone are included.

Milestone 1:

Be able to iterate character by character through your user entered string. You only need this working on one menu option, I recommend just focusing on the encode menu option right now. This makes sure you're able to interface with your string in a loop-like manner using recursion.

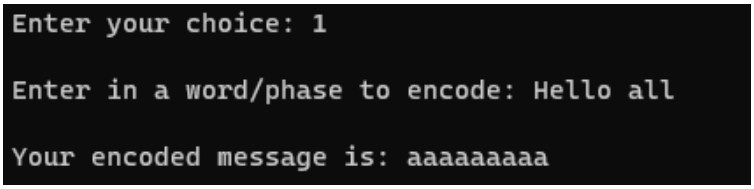
Must show:

- User input taken
- String the user has entered is displayed one letter at a time

Milestone 2:

Be able to alter your string, character by character. This doesn't have to be anything meaningful right now. Just pick a random character and see if you can change the entire word to be that character – this will mean you can do it once we actually implement the real encoding. My example will just show the character 'a' replacing whatever we enter in.

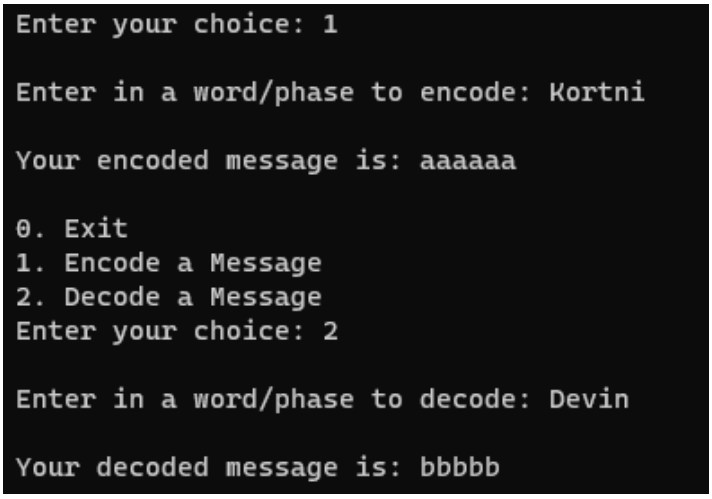
Must show:
- Input
- Input replaced with a character that matches the length of the word exactly



Milestone 3:

Respond differently based on encode true/false. Tack onto milestone 2 – replace with one character for option 1 and a different character for option 2.

Must show:
- Both menu options working
- Changes the word based on the menu option chosen



Milestone 4:

Add proper encoding. Must show:

- Properly ignoring non-alphabet characters
- Type sensitivity is retained
- The following encodings:
    - You’re a student.
        - Should be: Dtz'wj f xyzijsy.
    - Quick brown fox
        - Should be: Vznhp gwtbs ktc

Milestone 5:

Add proper decoding. Must show:

- Properly ignoring non-alphabet characters
- Type sensitivity is retained
- The following decodings:
    - Dtz'wj f xyzijsy.
        - Should be: You’re a student.
    - Ymj Frfetsnfs Ktwjxy
        - Should be: The Amazonian Forest

Hints / Tips:

Strings in C++ can be iterated through as if an array. They have indices, where each letter of the string is a C++ character. The size of a string can be obtained via the size function included in the string library.

As a reminder, characters in C++ are represented via integers from the ASCII table:

From the ASCII table...

Symbol	Decimal	Binary
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000110
G	71	01000111
H	72	01001000
I	73	01001001
J	74	01001010
K	75	01001011
L	76	01001100
M	77	01001101
N	78	01001110
O	79	01001111
P	80	01010000
Q	81	01010001
R	82	01010010
S	83	01010011
T	84	01010100
U	85	01010101
V	86	01010110
W	87	01010111
X	88	01011000
Y	89	01011001
Z	90	01011010

Symbol	Decimal	Binary
a	97	01100001
b	98	01100010
c	99	01100011
d	100	01100100
e	101	01100101
f	102	01100110
g	103	01100111
h	104	01101000
i	105	01101001
j	106	01101010
k	107	01101011
l	108	01101100
m	109	01101101
n	110	01101110
o	111	01101111
p	112	01110000
q	113	01110001
r	114	01110010
s	115	01110011
t	116	01110100
u	117	01110101
v	118	01110110
w	119	01110111
x	120	01111000
y	121	01111001
z	122	01111010

This means, you can very easily have something like this:

```
// this would store the letter E
char letter = ‘A’ + 4;

// this would store the letter R
char letter = ‘Z’ - 7;

// this is a totally valid comparison!
// it will compare the numbers associated with the chars
char letter = ‘R’;
if(letter < ‘Z’) … // rest of code
```

Keep in mind, towards the end of the alphabet, you’ll need to adjust characters to shift starting at A/a because the alphabet will need to wrap around. Also, take note that uppercase and lowercase letters have different number values associated with them.

Full Example Execution:

```
Welcome to a Caesar cipher simulator.

0. Exit
1. Encode a Message
2. Decode a Message
Enter your choice: 1

Enter in a word/phase to encode: Maggie Mae Neal

Your encoded message is: Rfllnj Rfj Sjfq

0. Exit
1. Encode a Message
2. Decode a Message
Enter your choice: 2

Enter in a word/phase to decode: Rfllnj Rfj Sjfq

Your decoded message is: Maggie Mae Neal

0. Exit
1. Encode a Message
2. Decode a Message
Enter your choice: 1

Enter in a word/phase to encode: Hello, my name is Kortni

Your encoded message is: Mjqqt, rd sfrj nx Ptwysn

0. Exit
1. Encode a Message
2. Decode a Message
Enter your choice: 0

Good-bye!
```

Comment Block:

Your code should contain a comment block at the top containing information on who wrote the code, what the assignment is, when it is due, etc. Here is an example of a good comment block to put:

```
/*
 Name: <your name>                          NetID: <your netID>
 Date: <current date>                       Due Date: <enter in due date>

 Description: <What is the program?>
*/
```

Deliverables:

- C++ code
    - Main file (.cpp file)

Point Breakdown:
(100 points total)
A submission that hasn't received a check-off will not be considered for grading.

Grading note:

A lab that achieves the goal without recursion will NOT be considered for a grade. It will be an automatic 0.

- 10pts - milestone 1
- 15pts - milestone 2
- 15pts - milestone 3
- 25pts - milestone 4
- 25pts - milestone 5
- 10pts - programming style *
- (PENALTY) 15pts - changing the code given to you (`main.cpp`)

* Programming style includes good commenting, variable nomenclature, good whitespace, etc.