Lab 1 (SGN Lab/Review)

Due Thursday at midnight

Time: This lab is a relatively straightforward lab. You should be able complete it in 3 hours or less. You should be able to do it in one sitting (as long as that sitting doesn’t start at 9:00 pm Thursday night!!!!!)

You may work on this during lab session. You may work with a partner on this lab (Be sure to submit your partner’s name on Canvas by Tuesday at 5:00pm!)

Contents

[Purpose of this lab: 1](#_Toc50321537)

[How this lab works: 2](#_Toc50321538)

[What you need to do: 2](#_Toc50321539)

[*For the lab you will need to do two things:* 2](#_Toc50321540)

[/\* Code \*/ 2](#_Toc50321541)

[Function Declarations (4 pts) 2](#_Toc50321542)

[/\*Function 1 (2 pts) \*/ 4](#_Toc50321543)

[\* How to print on the same line 4](#_Toc50321544)

[/\*Function 2 (2 pts) \*/ 5](#_Toc50321545)

[/\*Function 3 (3 pts) \*/ 5](#_Toc50321546)

[/\*Function 4 (4 pts) \*/ 5](#_Toc50321547)

[/\*Function 5 (4 pts) \*/ 6](#_Toc50321548)

[/\*Function 6 (3 pts) \*/ 6](#_Toc50321549)

[/\*Function 7 (3 pts) \*/ 7](#_Toc50321550)

[/\*Function 8 (4 pts) \*/ 7](#_Toc50321551)

[/\*Function 9 (4 pts) \*/ 7](#_Toc50321552)

[/\*Function 10 (5 pts) \*/ 8](#_Toc50321553)

[/\*Function 11 (4 pts) \*/ 8](#_Toc50321554)

[/\*Function 12a (3 pts) \*/ 8](#_Toc50321555)

[/\*Function 12b (5 pts) \*/ 9](#_Toc50321556)

[To Turn In: 9](#_Toc50321557)

## Purpose of this lab:

This lab should serve a few purposes:

1. Make sure your compiler is working properly
2. Get you familiar with C++ syntax and the differences between C and C++
3. Refresh your memory for coding
4. Get you used to my style
5. Ease you back into a crazy semester
6. Allow you to experience the joy and happiness of writing functions and getting your code to work

## How this lab works:

This lab is in essence a decoder lab. If you write the functions as described successfully, and run it with the main I’ve given you, you will decode the input I’ve given into intelligible output. Hint – if your output is gibberish, you did something wrong.

## What you need to do:

In eclipse, create a new project. Once you’ve created a project, create a new source file. If you are not familiar with how to do this, please watch my introductory videos on my web site (<https://www.eecis.udel.edu/~yarringt/CISC220> ). I go over how to create a project and a source file, and then how to compile.

Once you’ve created your source file (make sure it has a .cpp extension!), copy my code below.

## For the lab you will need to do two things:

1. Fill in the function declarations, above the main
2. Fill in the function definitions, as described in main, below the main.

(If you do not know the difference between the function declarations and the function definitions, I highly recommend you watch week 1’s videos on my web site, <https://www.eecis.udel.edu/~yarringt/CISC220> )

NOTE: DO NOT TOUCH THE MAIN FUNCTION!!!!! DID YOU HEAR ME? DON’T TOUCH IT. EVEN THOUGH YOU MIGHT WANT TO, RESIST THE TEMPTATION. YOU CAN DO IT!! JUST COPY AND PASTE THE WHOLE THING, ADD THE FUNCTION DECLARATIONS AT THE TOP, AND THEN BELOW THE MAIN FUNCTION, FILL IN THE FUNCTION DEFINITIONS AS DESCRIBED RIGHT ABOVE THEM. THAT’S IT. JUST STEP AWAY SLOWLY FROM THE MAIN FUNCTION. DON’T TOUCH IT.

(Okay, if you want to comment things out in main as you’re working on the different functions, and then remove the comments as you go, that’s fine. But my point is, you don’t have to write any code in the main function. It’s done for you.)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

## /\* Code \*/

**#include** <iostream>

**#include** <stdlib.h>

**#include** <string.h>

**using** **namespace** std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### Function Declarations (4 pts)

Fill in function declarations here

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**int** **main**() {

cout << "Hello World!" << **endl**;

string str0 = "ha";

**int** x = 12;

func1(str0,x);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string string1 = "ketxlurbaesrwainuts"; // 19 characters total

func2(string1,19);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string string2 = "psheskmarcaphulc";

func3(string2,16);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string str3 = "blek\_vpbq\_utxsreknmoihs\_neubr\_symluljiwsez\_aksivnzdw";

func4(str3,52,6,34);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string base1 = "woelihnbatmeudoirhdxkj";

**int** orig[] = {2,4,1,7,2,-2,3,1,3,7,-8};

func5(base1,22,orig,11,3);

**int** orig2[]={2,1,3,2,1,-2,8,-3,9,-1,3};

func5(base1,22,orig2,11,5);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### 

string s4 = "ekvmburwacvxezq\_ypotural\_ukinvnterzs\_powezikrdpvnbjesbsk!";

func6(s4, 57, 3, 4);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**char** arr5[] = {'a','s','x','j','t','i','h','o','q','e','p','w','v','t','z','m','i','h','y','m','k','d','i','c'};

**int** len = 24;

func7(arr5,len,7);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string s6 = "xezuhnbl\_uiplypdhqlb";

len = 20;

**int** arr6[] = {2351,92837,482,65,1039,233,23095,1,2037,693842,283};

**int** len2 = 11;

func8(s6,len,arr6,len2);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string s7 = "gboehdkixkwpmngzicosygmeuqlaltcvwdge";

len = 36;

func9(s7,len,6);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string s8 = "gboehdpixkwpangzicjsygmeaqlaltmvwdgeabndckswoshbk";

len = 49;

func10(s8,len,7);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string s9 ="w\_orp\_eshmueryasaa\_nootetkdtbueteei\_ars!vsfib\_t!e\_fslg\_!rye\_yra\_";

len = 64;

func11(s9,len,8);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### 

string somechars="rstlne";

string test = "arktole";

**for** (**int** i = 0; i < 7; i++) {

**if** (func12a(somechars,6,test[i],0)) {

cout << test[i] << " in charset"<<**endl**;

}

**else** {

cout << test[i] << " not in charset"<<**endl**;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string decipher = "bchzxivkjndpqsjiffrguxhjtb\_firskz\_azlckmjoxkrspktc\_ctprwepjznftjkyzxtkpwvueqnrqtckuyb!";

**int** len1 = 86;

string notchars = "bcfjkpqruvxz";

**int** len1a = 12;

func12b(decipher,len1,notchars,len1a,0);

**return** 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function Definitions \*/

/\* This is where you write the function definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 1 (2 pts) \*/

/\*Write a function that takes the string string0 and an integer

\* and prints that string the integer number of times. So if the

\* string was "glub" and the int was 5, the function would print out

\* >>glubglubglubglubglub (or, alternatively,

\* >>glub

\* glub

\* glub

\* glub

\* glub

\* (See note about how to make things print on the same line below)

\*

### \* How to print on the same line

\* Note: function declaration goes above main, function definition gets

\* written below main

\* Point: make sure you're comfortable with a basic loop

\* output on one line versus separate lines:

\* cout << "hi"<< endl;

\* pipes "hi" into a buffer, and the endl flushes the buffer and moves to

\* the next line. If you want to wait until all the characters are in the

\* buffer before you flush it and move to the next line, skip the endl;

\* so you'd have

\* cout << "hi";

\* and then whenever you want to flush the buffer, you would just add the

\* line,

\* cout << endl;

\* One final point: you can always add something after each thing that isn't

\* an endl;. YOu can add a comma:

\* cout << x << ",";

\* or a tab:

\* cout << x <<"\t";

\* or even a star:

\* cout << x <<"\*";

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **func1**(string s, **int** ct) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 2 (2 pts) \*/

/\* Write a function that takes the string below (string 1) and

\* passes it into func2,along with the length of the string. Using a while

\* loop, print out every other character.

\* The function func2 returns nothing.

\* Note: Write the function definition below the function, but include the

\* function declaration above main and below your using namespace std;

\* Point: I want to make sure you can loop through a string and that you

\* know how to use a while loop

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **func2**(string s, **int** len) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 3 (3 pts) \*/

/\*write a function that takes as input a string and the length of the string,

\* and converts the string to an array. The

\* array is not returned from the function. Using a for loop, print out every other

\* character in the array, starting at the last index in the array.

\* Point: I want to make sure you know how to create and traverse an array and how to use

\* for loop.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **func3**(string s, **int** len) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 4 (4 pts) \*/

/\* write a function that takes as input a string, the length of the string, the index

\* of where you're starting to reverse the string, and the size of the area to be

\* reversed. The function returns nothing.

\* The function should convert the string to an array (because strings are

\* immutable!!! Can't change 'em!!). It should then reverse, within the array,

\* the values between the index and the size.

\* So, for instance, if you had,

\* {3,2,7,5,9,1,2,8,4} and were told to, starting at index 2, reverse size of 5,

\* the resulting array would be:

\* {3,2, 2,1,9,5,7, 8,4} //I left the spaces to make it a bit clearer.

\* Then print out every other value.

\* Looking at indexing and looping

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **func4**(string str,**int** len,**int** ind1,**int** size) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 5 (4 pts) \*/

/\*write a function that takes as input a string, the length of the string, an

\* array of ints, and the length of the array. Finally, the function should

\* take a filter size. (The filter size can be any positive number).

\* For the first call, the filter size will be 3.

\* Inside the function, find the sum of the integer array over the filter size.

\* Use that summed number as an index into your base to create a word.

\* E.g., if the base was, "stwcea", the int array was, "1,0,2,3,-4", and the

\* filter size was 3, the summed array would be: 3,5,1 (notice that it is

\* 2 shorter), and then using those values as indices into the base, you'd

\* get "cat"

\* Please note that the function should be written to take ANY filter size!

\* I'm looking at multiple loops and summations

\*/

**void** **func5**(string s, **int** len, **int** orig[],**int** size, **int** fsize) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 6 (3 pts) \*/

/\*write a function that takes as input a string, the length of the string,

\* and 2 integers (x and y). The function then loops through the string,

\* printing out every character in the string whose index is evenly divisible

\* by either x or y, but not both.

\* Looking at if conditions, truth conditions

\*/

**void** **func6**(string s, **int** len, **int** x, **int** y) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 7 (3 pts) \*/

/\*write a function that takes as input an array of chars, the length of the array,

\* an integer rot. In the function, rotate the characters in the array to the left

\* by whatever number rot is. (NOTE: DO NOT CREATE A NEW ARRAY!)

\* So, for instance, if the string was,

\* "abcdefghij"

\* and rot was 4

\* The resulting string would be:

\* efghijabcd

\* Then print every third character in the array

\* Point: Looking at array manipulation and temp values

\*/

**void** **func7**(**char** arr[], **int** len, **int** rot) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 8 (4 pts) \*/

/\* write a function that takes as input a string, the length of the string,

\* an array of numbers, and the length of the array of numbers.

\* For each of the numbers in the array of numbers, the function adds up the

\* digits in the number. It then calculates an index into the array by

\* finding the remainder when dividing by the string size.

\* So if the string is:

\* "grandma"

\* the length of the string is 7.

\* Say the number in the array of numbers is 5497

\* Adding the digits results in 25.

\* If we divide by 7, the remainder is 3.

\* So the resulting character would be 'n' (which is at index 3 in the string)

\* Point: number manipulation

\*/

**void** **func8**(string s, **int** len, **int** arr[], **int** len2) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 9 (4 pts) \*/

/\* write a function that takes as input a string, the length of the string,

\* and a matrix dimension size.

\* The function creates a matrix of size by size and fills in the characters

\* such that the first size characters are in the first row, the second size

\* characters are in the second row, etc.

\* Then the function should print out the diagonal characters (0,0 to size,size).

\* Point: creating and accessing a matrix

\*/

**void** **func9**(string s7,**int** len,**int** msize) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 10 (5 pts) \*/

/\* write a function that takes as input a string, the length of the string,

\* and a matrix dimension size.

\* The function creates a matrix of size by size and fills in the characters

\* such that the first size characters are in the first row, the second size

\* characters are in the second row, etc.(so far just like the function above)

\* (Then the function should print out the reverse diagonal characters

\* Top right corner to bottom left corner)

\* Point: creating and accessing a matrix

\*/

**void** **func10**(string s,**int** len,**int** msize){

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 11 (4 pts) \*/

/\* write a function that takes as input a string, the length of the string,

\* and a matrix dimension size.

\* The function creates a matrix of size by size and fills in the characters

\* such that the first size characters are in the first row, the second size

\* characters are in the second row, etc.(so far just like the functions above)

\* (Then the function should print out the reverse diagonal characters

\* Then the function should print the columns from top to bottom. So, for

\* instance, if the string was,

\* "abcdefghi" and the matrix size was 3, the resulting matrix would be:

\* abc

\* def

\* ghi

\* and what would be printed out would be:

\* adgbehcfi

\*Point: accessing matrix

\*/

**void** **func11**(string s,**int** len,**int** msize) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 12a (3 pts) \*/

/\* This part has 2 functions: 1 function that returns a boolean value,

\* and one function that returns a string.

\* BOth must be recursive.

\* Note: if you need a refresher on writing recursion, there is a

\* tutorial on my web site (<https://www.eecis.udel.edu/~yarringt/CISC220> )

\* under student info.

\* Part 1:

\* Write a recursive function that takes as input a string, a character, and an int.

\* The function returns true if the character is in the string, and false

\* otherwise.

\*Point: well, recursion

\*/

**bool** **func12a**(string somechars,**int** len, **char** c,**int** ind) {

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### /\*Function 12b (5 pts) \*/

/\* Part B:

\* Now write a recursive function that takes as input a string, a length of

\* the string, a second string, a second length, and an index; It iterates

\* through the first string and, using the second string and the above

\* (recursive function) it checks to see if a character is in the second string.

\* If it is not, it prints it out. Otherwise it skips that character.

\* So, for instance, if the two input strings were,

\* "acokpdae" and

\* "vtmipksa"

\* a is in the second string, would not be printed,

\* c is not in the second string, would be printed

\* >>c

\* o is not in the second string, would be printed

\* >>o

\* k is in the second string, would not be printed

\* p is in the second string, would not be printed

\* d is not in the second string, would be printed

\* d

\* a is in the second string, would not be printed

\* e is not in the second string, would be printed:

\* so the result would be:

\* >>c o d e

\* Point: again, recursion.

**void** **func12b**(string decipher,**int** len1,string notchars,**int** len2,**int** ind){

/\* write function definition here \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

## To Turn In:

1. Make sure your code compiles
2. Make sure your partner’s name is on the program, along with your own
3. Take a screenshot of the working code results\*
4. Zip (compress) your .cpp file and your screenshot (this is to get you into the habit for when you’re submitting multiple files simultaneously)
5. Submit your zipped file on canvas Thursday night before midnight.

*\*Why? Because if for some reason the program you wrote compiles and runs on your computer, but it does not compile or run on the TA’s computer, with a screenshot of the running code, we will make an appointment to see your code run rather than just give you a 0. This happens when we’re using different compilers sometimes (even with MacOSX vs MINGW compiler!)*