CISP 400 – Lab Assignment #2
Due Wednesday, September 12 at 11:59 PM

The purpose of this lab is to run you through several exercises involving common C and C++ programming techniques. It is important that you understand the concepts presented here. When a part of the program specification tells you to use a specific technique, use that technique; other techniques may receive partial credit, or no credit.

This lab covers:

- Basic I/O
- Variables and data types
- Operators
- Expressions
- Control structures (loops and decisions)
- Type casting
- Functions
- Arrays
- Pointers (limited)

The assignment is writing a program that puts together a simple menu for a user at a text-based terminal. The user will be prompted for input at various times, such as the menu, and other times when we are doing math of one sort or another.

Main Function:

The main function will be a menu. It will give a listing of the different functions that are available, giving a number to each. It will invite the user to choose one, or to exit the program. If the user picks a number that is NOT available on the menu, the program will tell the user that they made a mistake, and ask them again. If the user picks a valid menu item, that function runs, and then they are asked to choose again. If the user picks the menu item for exit, the menu loop breaks and the program reaches the end of the main() function, ending the program entirely.

Subtraction:

This function, called subtractNumbers(), will take two FLOATING POINT numbers from the user as input. It will then pass these values to another function (call it calculateSubtraction()). calculateSubtraction() will subtract the numbers and return the result. Once calculateSubtraction() is done, subtractNumbers() will print out the result and exit.

Adding Array Contents Up:

This function, called countingArray(), should have an array of 10 integer elements declared. It begins by initializing all of the values in the array to -1. It should then ask the user whether or not they want to enter a variable (ask them for a Y/N response), which will repeatedly ask the question if the user does not provide a Y or N (say, responds with a ‘q’). If the user chooses ‘Y’, then the user is allowed to enter an integer. That integer is then placed in the array. This continues until either the user enters 10 values, or the user replies that they do not want to enter any more. Note that you do NOT

need to check for both an uppercase Y or N and a lowercase y or n; you can if you want to, but just lowercase is fine.

The parts of the array that are used will then be added together with a loop, totaling them. Remember, your program will have to KEEP COUNT of how many of the slots have been used, both to place them correctly and to do this addition operation! The total will be printed to the terminal and the function will exit.

Multiplication Table:

This function, multiplicationTable(), will use a two-dimensional int array, specifying that it is 10 by 10. It will then loop and build a multiplication table, by multiplying the two variables being used to keep track of where it is in the array.

It will then print the array with another set of loops. The function will then exit.

Casting:

This function, called castToShort(), will prompt the user to enter a floating-point number, specifically a variable of type **double**. Then, using a typecast, the program will then take the floating-point number and put it into a **short** variable. The assignment **must** use **static_cast** as was demonstrated in class. The program will then present the integer value from the floating-point variable, and then exit.

Countdown:

This function, called runCountdown(), will declare an array of 20 elements. It will pass this array to another function, named arrayCountdown(). The function prototype for arrayCountdown() is given here:

```
void arrayCountdown(int *outputArray, int elements);
```

The parameter outputArray is the destination for the work of the arrayCountdown() function. The elements parameter is the number of elements to fill.

arrayCountdown() will use outputArray and number of elements to run a counting loop. The loop will place the values 1 through the number of elements provided in the array. Remember the fact that arrays start counting at 0, so array element 0 will hold the value 1, array element 1 will hold the value 2, etc.

Once arrayCountdown() has completed its work, runCountdown() will run a loop that will print out the entire value of the array that was filled by arrayCountdown().

Turning In:

The complete program should be turned in as source code to Canvas by the due date. Remember, the extension for C++ programs is “.cpp”.