This lab is a refresher for use of arrays and to practice compiling and running your programs in both Windows AND Linux.

Note: In order to complete the following two assignments, you need to create your own main() to test the correctness of your functions.

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

Submit TWO documents (each program has its own document to turn in) to Canvas. You will need to include two sets of both executed test plans and output snippets of test results (one each for both Windows and Linux) for each document. Test plan template can be found in files.

i.e. Document 1: Windows source code; Windows completed test plan; output snippets of Windows test execution; Linux completed test plan; output snippets of Linux test execution.

**RUN IN BOTH LINUX AND WINDOWS VISUAL STUDIO**

Program 1

Given that arrayIntValues [MAX\_ROWS][MAX\_COLUMNS] is a 2 dimensional array of positive integers, write a C++ function *howManyEven* to find the total number of even elements in the array. It should have input parameter array arrayIntValues. The function should return an integer. Also create a C++ subroutine called *printArray,* with the input parameter array arrayIntValues, to print out the elements in the array (be sure to have row and column labels and proper formatting (i.e. column values line up). Main must be in one source file and all other functions in a second source file.

The constants MAX\_ROWS and MAX\_COLUMNS must be accessed globally by defining the following global variables:

const int MAX\_ROWS = 3; or #define MAX\_ROWS 3

const int MAX\_COLUMNS = 2; or #define MAX\_COLUMNS 2

The array arrayIntValues is initialized by the following statement within main():

int arrayIntValues [MAX\_ROWS][ MAX\_COLUMNS] = { {3 , 2}, {4, 5}, {2, 2} };

Print out the array and the count of positive even numbers (label appropriately) in function *printArray*.

User enters a new set of values into the array (must do it at least once) as many times as they wish and runs *printArray* against the new array values each time. Be sure to validate input is positive integers.

Validate user input (“y” or “n”) when asking if the user wants to enter a new set of values.

Program 2

Given storeMonthlySales[NUM\_STORES][NUM\_MONTHS][NUM\_DEPTS] is a three-dimensional array of floating point values. Write a C++ function, *printMonthlySales*, to calculate and print the total value of sales during a specific *month* by each department and in each store plus totals by store and department and overall sales. The return type of the function is void and the function must have 2 input parameters: Sales array and the *month* the user specified (edit for correct month values!). The constants NUM\_STORES, NUM\_MONTHS, and NUM\_DEPTS must be accessed globally by defining the following global variables:

#define NUM\_DEPTS 2 or const in NUM\_DEPTS = 2;

#define NUM\_STORES 2 or const int NUM\_STORES = 2;

#define NUM\_MONTHS 12 or const int NUM\_MONTHS = 12;

The array Sales is initialized by the following statement within main():

float storeMonthlySales[NUM\_STORES][NUM\_MONTHS][NUM\_DEPTS] =

{ 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2,

2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2,

3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 4.0, 4.1, 4.2,

2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2

};

Output: Sales for month of value (where value represents “January”, “March”, etc.)

Dept # Dept # Store Total

Store # value value row value sum

Store # value value row value sum

Dept Total col value sum col value sum total overall sales

User enters a new *month* as many times as they wish and run *printMonthlySales* against the new month value.

Validate user input (“y” or “n”) when asking if the user wants to enter a new set of values.