**ECE 1111: Engineering Computation I**

**Laboratory No. 8: Arrays and Strings**

**Goals:** Help you develop an understanding of how to do basic linear algebra. Also, we are introducing formal design and verification steps into this lab.

**Deliverables:**

1. *Diagram:* explain how "char\*\* argv" is organized in memory using a diagram. Explain how this is both similar and different to a traditional numeric vector. (directory: p01; filename: diagram.pdf). In your diagram, you should show specific memory addresses and the values contained at those addresses.
2. *Source Code:* a program that multiplies two matrices. The program’s name should be *p02.cc, and the binary should be p02.exe.* It should compile with the make command. (directory: p02; main program: p02.cc).
3. *Check Off:* Demonstrate that your program runs successfully and produces the same result as Excel. (directory: p03; filename: p03.xlsx).

Deposit your work in:

/data/courses/ece\_1111/current/labs/lab\_08/<lastname\_firstname>

**Description:**

In this lab, we will introduce you to some simple operations involving matrices. We will use the file I/O tools you developed in previous assignments. All computations, including I/O, are to be done using floats (32-bit floating point numbers). The specific tasks are:

1. Using the debugger (gdb) to demonstrate how "argv[1]" is organized in memory. Compile your code debug using the “-g” option. Set a breakpoint in the main program. Run the program with the following command line arguments:

p02.exe Jordan Alex “see Alex run”

Show where these arguments reside in physical memory. Show how the values you observe map to the diagram you prepared. Walk your TA through each component for argv and show where it is located in memory in an actual program. (use *p02.cc*).

1. Write a program that loads two matrices from a text file and multiplies them. Specify a dimension followed by the values of the matrices. Test your code using identity matrices. Then demonstrate that your program works for arbitrary matrices as long as they are the same dimension. Verify your answers using Excel.

For (2), a simple test case would be an identity matrix:

nedc\_000\_[1]: more m.txt

3 3

1.0 0.0 0.0

0.0 1.0 0.0

0.0 0.0 1.0

3 3

1.0 0.0 0.0

0.0 1.0 0.0

0.0 0.0 1.0

The interface to your program should be:

myprog.exe <filename>

The output of your program should be a nicely formatted display of the inputs and the outputs. Note that matrix multiplication of a matrix with an identity matrix produces the input matrix, so you can use the above example for the initial set of test cases to debug your code.

Here is another simple test case:

Input 1: rows = 3, cols = 3

0: 1.0 0.0 1.0

1: 0.0 1.0 0.0

2: 0.0 0.0 1.0

Input 2: rows = 3, cols = 3

0: 1.0 0.0 1.0

1: 0.0 1.0 0.0

2: 0.0 0.0 1.0

Matrix Multiplication Result: rows = 3, cols = 3

0: 1.0 0.0 2.0

1: 0.0 1.0 0.0

2: 0.0 0.0 1.0

Your program should work for any size matrix. It should produce an error message if the matrices are not the same dimension. Your inputs and outputs should be properly formatted. Remember to comment your code and properly format it.

You will probably find this tutorial helpful with your I/O: [*https://www.programmingsimplified.com/c-program-add-matrices*](https://www.programmingsimplified.com/c-program-add-matrices).

**Summary:**

Vectors, or an array as we often refer to it, and matrices (technically simply a multidimensional array) are fundamental to the way we write code to implement engineering mathematics. In this lab, we cover the basics of how these types are stored in memory. Once we formally introduce pointers, array representations will make much more sense.