**ECE 1111: Engineering Computation I**

**Laboratory No. 6: Simple Engineering Mathematics**

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

**Deliverables:**

1. *Flowchart:* a block diagram of how your code works. Submit this as a pdf file.
2. *Source Code:* C/C++ and an Excel file -or- Matlab program that computes the equation.
3. *Check Off:* Demonstrate that your program runs successfully and produces the same result as Excel or Matlab. Explain how your code implements the flow chart you have developed.

Deposit your work in:

/data/courses/ece\_1111/current/labs/lab\_06/<lastname\_firstname>

**Description:**

In this lab, we will integrate several concepts we have discussed in class and demonstrated during the lectures or labs. The specific task is to implement a program that reads an array of data from a file into a vector, computes the following equation:

and prints the result to stdout. and are your input vectors, "" denotes a vector dot product, is a constant, and is the result (a scalar or number).

Create two text files that contains the length of the vector followed by the elements of the vector. For example, a simple test case might be:

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

3

1.0

0.0

-1.0

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

3

0.0

1.0

0.0

The interface to your program should be:

myprog.exe <filename1> <filename2> c

where the first two arguments are the filenames and the third argument is the constant in the above equation. Your source code should be in a file called *myprog.cc* and your binary should be *myprog.exe*.

The output of your program should be:

the dimension of the vector is <dimension>

z = <result>

If the files fail to open properly, print an error message and exit. If the vectors are not the same lengths, print an error message and exit. If the dimension does not match the data in the file, print an error message and exit.

Print the result using four decimal places of precision. Demonstrate that your code produces the same result as Excel or Matlab by implementing this in either.

Use the atof() function to convert the third command line argument to a value inside your program. Also use the atof() function to convert the ASCII text in your file to a number that can be assigned to each element of your vector.

Though we haven't formally covered vectors yet, the code below demonstrates all you need to know for this lab with respect to vectors:

long N = 27;

float x[N];

for (long i = 0; i < N; i++) {

fprintf(stdout, "x[%d] = %10.4f\n", i, x[i]);

}

Try Google searching terms like "how do I declare a vector in C" or "how do I index a vector in C" to learn more about vectors.

Before you write your code, you need to develop a block diagram of how your code works. This is a process known as flowcharting. See *https://www.lucidchart.com/pages/what-is-a-flowchart-tutorial* for an explanation of what is a flowchart. You can use whatever tool you like to generate this, including the block diagram builder in PowerPoint. Generate a pdf containing your final chart.

Test Case: for the example shown previously, the dot product should be zero. Why?

**Summary:**

This lab demonstrates several key concepts in programming: (1) data-driven programming: reading data from a file rather than hardcoding it in your program; (2) dynamic memory allocation: sizing your program to the data so you don't run out of memory; (3) flowcharting: a visual representation of the design of your program; and (4) verification: using MATLAB or Excel to verify your results.

Design is perhaps the most important part of software development. There are many ways we design code, including the use of rapid-prototyping tools such as Excel or MATLAB. Flowcharts are a convenient way to visualize the structure of your code. You should always do a formal design of your code before you start implementing it so that you have a good idea of how the code needs to be structured.

Make sure you use a make file, a header file and that you format and comment your code as per the course requirements.