**ECE 1111: Engineering Computation I**

**Homework No. 3: Basic Math Operations and Numerical Precision**

**Goal:** Demonstrate that we must be cognizant of numerical precision when programming. Electrical and computer engineers are expected to be able to write efficient code for embedded systems, even if you are working in a high-level language. We are expected to write code for systems that run perpetually. Roundoff errors can accumulate over time and cause these systems to fail after billions of operations. In this assignment, you will gain some experience with numerical precision issues.

**Description:** In this homework assignment, we are going to use a loop of the following form:

**long i\_end = 999;**

**for (long i = 0; i < i\_end; i++) {**

**fprintf(stdout, "the value of i is: %d\n", i);**

**}**

You will also find the following command useful: cat filename.txt | more. Piping to "more" sends the output of your program to a program called more that lets you control the output. In addition to "more", there are commands called "less" and "tail". Experiment with these.

Place your files in the directory:

**/data/courses/ece\_1111/2019\_fall/homework/hw\_03/lastname\_firstname**

Use subdirectories p01 and p02 for the problems below.

To make things easy, place the following include statement at the top of your C program:

**#include <math.h>**

and compile and link your program using this command:

**gcc -o myprog.exe -lm myprog.cc**

The tasks in this homework assignment are:

1. Declare an unsigned character, which is an 8-bit (1-byte) variable. Increment its value using the code below. Explain what happens and why there might be a problem.

 **unsigned char c = 0;**

 **for (long i = 0; i < 99999; i++) {**

 **fprintf(stdout, "c = %c (%d)\n", c, (long)c);**

 **c++;**

 **}**

Repeat this for an unsigned short int, an unsigned int and an unsigned long.

Change this loop to iterate from -99999 to 99999. Repeat the above for signed char, an unsigned short int, a signed short int, an int and a long. Explain what you are observing.

1. Declare a floating-point value for the math constant pi:

**float my\_pi = M\_PI;**

Construct a loop that sums the square of M\_PI 99,999 times. Divide the sum by 99,999 and print the difference between value computed and the theoretical value (M\_PI \* M\_PI). Use a format of %15.10. What do you observe? How does the result change if you decrease 99,999 to 999, or increase it to 9 million?

Repeat this for a double instead of a float. Does the result change? Why? Explain.

In addition to your code, submit the solutions to these tasks as a pdf document using a filename of lastname\_firstname\_hw03.pdf following the homework template provided. Place this in the parent directory.