**CS 3723 Programming Languages**

Assignment #4 – Call by value, call by reference, call by name, etc.

You have been given 7 C files, each has the same logic, expressed in different ways. Each of the 7 is incomplete, marked by TBD. You must replace each TBD entry with appropriate C code. You must NOT alter any of the existing lines. You are only allowed to replace each TBD line with one or two new lines.

Each of the 7 C programs do exactly the same task. Given a list of numbers on the command line, they find the smallest positive number and the largest negative number. You do not have to add error checking. You can safely assume (1) all numbers are given nicely as integers on the command line, (2) both positive and negative numbers will always be present, and (3) all numbers will be in the range of -1,000 to +1,000.

The output from all 7 will be the same for the same arguments. For example, the smallest positive number in the list (34 82 -4 -22 13 -83 0 3) is 3, and the largest negative number is -4.

The 'doit.sh' script is available to use if you choose. It runs on Linux and will compile and run each of the 7 C programs for you. The script is optional and does not need to be submitted.

## Part I: Implementation (5 points each, 35 points).

Get each of the 7 C files to compile and produce the results as given above. Submit your code (and output) in Blackboard. Please submit a single zip file, called Lastname\_abc123.zip, with 7 C files and a transcript called mmm.out which shows the output from all 7 executables.

## Part II: Discussion (5 points each, 25 points).

1. Which version(s) are unsafe when multi-threaded? mmm2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Why? \_The values are stored in static global variables, which means they can technically be modified by multiple threads at the same time

1. Which version is the hardest to read and maintain? \_mmm2\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Why? \_The global variables can be changed by any function calls without knowing exactly what was done or where. This can make for a huge headache down the road as the program grows or changes.

1. Which version has the most redundant code? \_mmm1\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Why? \_There is a lot of boilerplate code in each function that could be de-duplicated if they were combined

1. Which version(s) are closest to call-by-name? \_\_mmm7\_\_\_\_\_\_\_\_\_\_\_

Why? \_This version expands the function call to the code in the definition, which replaces the variables in the macro with the arguments passed. This means the names of the arguments are evaluated in the calling environment, which is how call by name works.

1. Which version(s) are closest to call-by-reference? \_mmm4, mmm6\_\_\_\_\_\_\_

Why? \_In these versions, pointers to variables or structures are passed to the function, and then they are used to change the variables in the calling function, similar to what would happen in a call by reference language.

## Bonus: Implement in Python (5 points).

You must calculate the smallest positive and largest negative numbers. You only need to implement one version, and submit one mmm.py file. A partial solution has been provided for you. As above, you can only replace TBD lines.