EECE5640
High Performance Computing
Homework 3
*Submit your work on Turnitin on Blackboard

1. (20) In this problem, you will utilize IEEE 754 format and consider the performance implications of using floats versus doubles in a computation.

a.) Compute $f(x) = x + e^x$ using a Taylor series for e^x. To refresh your memory, $e^x = 1 + x/1! + x^{2}/2! + x^3/3! + \ldots..$ You select how many terms you want to compute (but at least 10 terms). Compute f(x) for 3 different values, but be careful not to use too large a value. Generate two versions of your code, first defining x to be a float, and second defining x to be a double. Discuss any differences you find in your result of f(x). You do not need to provide your code, but you should provide an in-depth discussion on the results you get, and the reasons for any differences.

b.) Provide both IEEE 754 single and double precision representations for the following numbers: 1.0, 6400, and -0.005.

*Answers to this question should be included in your homework write-up in pdf format and submitted through Turnitin. Just include the code in your writeup.

2. (20) Using Linux semaphores and Posix pthreads, implement a C/C++ program that counts the number of numbers between 1-5000 that are evenly divisible by 3 or 5, or by both. You will use 8 threads, distributing the work to the 8 threads to perform both checking. At the end, you will print out the number of numbers that were divisible. Also print out the number values that were found. Make sure you use pthreads for generating your 8 threads, and semaphores for synchronization.

* You should include a C/C++ program submitted through Turnitin.

3. (20) Revisit the Dining Philosophers Problem, but this time develop your solution in OpenMP. You will then answer the following questions:

a.) Compare and contrast how difficult/easy it was for you to leverage OpenMP, as compared to pthreads, for your implementation.

b.) Compare the performance of the two implementations – you will probably need to modify your program to get accurate performance results. Do not just provide performance numbers, try to explain any differences you see your results.

*Answers to this question should be included in your homework write-up in pdf format, and submitted through Turnitin. You should include a C/C++ program submitted through Turnitin.

4. (20) Write an OpenMP program that performs a matrix-vector multiply on an 512x512 matrix of integers, multiplied by a 512x1 vector. Initialize the matrix in your program. Then have your program:
   a.) Print out the number of processors available to run the program on a multicore system of your choice and print out a unique threadID for each thread using the OpenMP built-in function.
   b.) Print out the resulting vector.
   c.) Use the OpenMP built-in reduction to compute each element of the output vector.

   *Answers to this question should be included in your homework write-up in pdf format, and submitted through Turnitin. You should include a C/C++ program submitted through Turnitin.

5. (20) (extra credit for undergraduates and BS/MS students, but required for graduate students) IWOMP is an international workshop that focuses on innovations in OpenMP research. Select a paper from a recent year of the workshop, and write a summary of the research being reported.

   *Answers to this question should be included in your homework write-up in pdf format, and submitted through Turnitin.