EECE5640
High Performance Computing
Homework 2
***Submit your work on Canvas in a single zip file.**

1. (50) In 1965, Edsger W. Dijkstra described the following problem. Five philosophers sit at a round table with bowls of noodles. Forks are placed between each pair of adjacent philosophers. Each philosopher must alternately think or eat. However, a philosopher can only eat noodles when they have both left and right forks. Each fork can be held by only one philosopher, and each fork is picked up sequentially. A philosopher can use the fork only if it is not being used by another philosopher. Eating takes a random amount of time for each philosopher. After a philosopher finishes eating, the philosopher needs to put down both forks so they become available to others. A philosopher can take the fork on their right or the one on their left as they become available but cannot start eating before getting both forks. Eating is not limited by the remaining amounts of noodles or stomach space; an infinite supply and an infinite demand are assumed.

   Implement a solution for an unbounded odd number of philosophers, where each philosopher is implemented as a thread, and the forks are the synchronizations needed between them. Develop this threaded program in pthreads. The program takes as an input parameter the number of philosophers. The program needs to print out the state of the table (philosophers and forks) – the format is up to you.

   Answer the following questions: you are not required to implement a working solution to the 3 questions below.

   a.) Does the number of philosophers impact your solution in any way? How about if only 3 forks are placed in the center of the table, but each philosopher still needs to acquire 2 forks to eat?
   b.) What happens to your solution if we give one philosopher higher priority over the other philosophers?
   c.) What happens to your solution if the philosophers change which fork is acquired first (i.e., the fork on the left or the right) on each pair of requests?

   When you submit this portion of the assignment, provide clear directions on how you tested your code so that the TA can confirm that your implementation is working. Provide these directions in a README file which instructs how to run through at least 12 iterations of updating the state of the philosophers and forks around the table.

   For 5 points of extra credit, discuss Edgar Dijkstra in your write-up, describing who he was and what is so important about this dining problem, as it relates to the real world. Make sure to discuss the algorithm that bears his name, Dijkstra's Algorithm. Cite your sources carefully.

*Written answers to the questions should be included in your homework 2 write-up in pdf format. You should include your C/C++ program and the README file in the zip file submitted.

2. (30 MS/PHD, 50 BS/PlusOne) In this problem you will develop two different implementations of the computation of **`pi`** numerically using pthreads and OpenMP. Then you will compare them in terms of scalability. **Undergraduates only need to complete parts b and c of this problem for full credit, but can complete part a for 10 points of extra quiz credit.**

In this problem, you will develop a program that computes the value of pi. You can refer to the following video that suggests a way to compute this using Monte Carlo simulation:

`https://www.youtube.com/watch?v=M34TO71SKGk&ab_channel=PhysicsGirl`

This may not be the most efficient algorithm. There may better ways to compute the sequence. If you do choose a different method, compare it to the Monte Carlo method in terms of convergence rate (assessing the accuracy of the value as a function of runtime).

a. Evaluate the speedup that you achieve by using pthreads and multiple cores. You should vary the number of threads. The program should take two input parameters, the number of threads and the number of "darts" thrown. Your program should print out the time required to compute pi and the final value of **`pi`**. Make sure to document the system you are running on and the number of hardware threads available. Try to explain the trends you are seeing.

b. Now develop the same program using OpenMP. Repeat all of the steps requested in part a.).

c. Now compare the two implementations in terms of strong and weak scaling, where the number of Monte Carlo simulations (i.e., "darts" thrown) is used to assess weak scaling.

   * Written answers to the questions should be included in your homework 2 write-up in pdf format. You should include your C/C++ program and the README file in the zip file submitted.

3. (20 MS/PhD, 20 points of extra quiz credit BS/PlusOne) Read the paper by Castelló et al. that considers lightweight threads versus sticking with operating system threads. Then answer the following questions:

   a.) Discuss some of the tradeoffs of using OpenMP alone versus adding the capabilities of light-weight threading libraries described in the paper.

   b.) Select one of the lightweight threading libraries discussed and provide an example of how you would use it to parallelize a simple vector addition kernel (adding two vectors of single-precision floating point numbers together). You do not need to provide code that compiles, just provide enough syntax to show you know how to use the library.

c.) The paper evaluates the performance of BLAS-1 functions. Discuss what is included in the BLAS-1 subprograms.

*Answers to this question should be included in your homework write-up in pdf. Make sure to cite your sources.

4. (20 points of extra credit for all) Utilize semaphores instead of mutexes in your solution to problem 2a.

   * Written answers to the questions should be included in your homework 2 write-up in pdf format. You should include your C/C++ program and the README file in the zip file submitted.