# Assignment 1

ECSE 420: Parallel Computing

*Due: October 12, 2017*

## Submission instructions: Students are to work in groups of two on the assignment.

Students must also submit a pdf file that contains the following information: student's names, student ids, instructions on how to run each file and the associated question solved. Students are also expected to submit a zip file containing their source code. This code must compile and run without error. Code must be well formatted, commented, and follow the google java style guide. For more information, see the ECSE420AssignmentSubmissionInstructions.pdf in the Assignment section on myCourses.

## Questions

1. (24 marks) Matrix multiplication is a common operation in linear algebra. Suppose you have multiple processors, so you can speed up a given matrix multiplication.
   1.1. Implement a matrix multiplication method sequentially with the following signature: *public static double[][] sequentialMultiplyMatrix(double[][] a, double[][] b)*
   1.2. Implement a matrix multiplication method in parallel with the following signature. *public static double[][] parallelMultiplyMatrix(double[][] a, double[][] b)*
   1.3. Write a test program that measures the execution time for multiplying two 2,000 by 2,000 matrices using the parallel method and sequential method, respectively. Be sure to indicate how many threads are being used.
   1.4. Vary the number of threads being used by the parallel method for the same size multiplication as in 1.3 and plot execution time as a function of the number of threads. Be sure to indicate the size of the array used.
   1.5. Vary the size of the matrix being multiplied and plot the execution time as a function of matrix size for the parallel and sequential methods respectively. Be sure to indicate how many threads are being used in the parallelization.
   1.6. For the generated graphs in 1.4 and 1.5 comment on their shape and possible reasons for the observed behavior.

2. (8 marks) Write a program that demonstrates deadlock.
   2.1. Comment on the consequences of deadlock
   2.2. Discuss possible design solutions to avoid deadlock

3. (16 marks) The dining philosopher's problem was invented by E. W. Dijkstra, a concurrency pioneer, to clarify the notions of deadlock and starvation freedom. Imagine five philosophers who spend their lives just thinking and feasting. They sit around a circular table with five chairs. The table has a big plate of rice. However, there are only five chopsticks (in the original formulation forks) available, as shown in Fig. 1. Each philosopher thinks. When he gets hungry, he sits down and picks up the two chopsticks that are closest to him. If a

philosopher can pick up both chopsticks, he can eat for a while. After a philosopher finishes eating, he puts down the chopsticks and again starts to think.

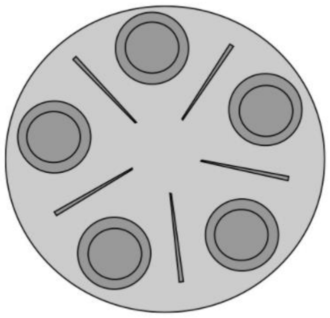

*Figure 1*

3.1. Write a program to simulate the behavior of the philosophers, where each philosopher is a thread and the chopsticks are shared objects. Notice that you must prevent a situation where two philosophers hold the same chopstick at the same time.
3.2. Amend your program so that it never reaches a state where philosophers are deadlocked, that is, it is never the case that each philosopher holds one chop- stick and is stuck waiting for another to get the second chopstick.
3.3. Amend your program so that no philosopher ever starves.
3.4. Write a program to provide a starvation-free solution for any number of philosophers n.

4. (12 marks) Use Amdahl's law to answer the following questions:
   4.1. Suppose the sequential part of a program accounts for 40% of the program's execution time on a single processor. Find a limit for the overall speedup that can be achieved by running the program on a multiprocessor machine.
   4.2. Now suppose the sequential part accounts for 30% of the program's computation time. Let $s_n$ be the program's speedup on n processes, assuming the rest of the program is perfectly parallelizable. Your boss tells you to double this speedup: the revised program should have speedup $s'_n > s_n*2$. You advertise for a programmer to replace the sequential part with an improved version that runs $k$ times faster. What value of $k$ should you require?
   4.3. Suppose the sequential part can be sped up three-fold, and when we do so, the modified program takes half the time of the original on $n$ processors. What fraction of the overall execution time did the sequential part account for? Express your answer as a function of $n$.

Total: 60 marks