EEL 5934/4930 Advanced Systems Programming
Assignment 3
Due: Monday, February 22nd by midnight

In this assignment you are going to use Pthreads library to implement a semaphore-based and deadlock-free solution to the Dining Philosophers. In the Dining Philosophers problem there are N philosophers seated on a circular table and there are N forks, which are also placed in a circular fashion so that each philosopher has one on its right and one on its left. A philosopher needs two forks to eat. Your solution should enable synchronization of processes, i.e., each philosopher will run as a separate process. To achieve this, you will use Pthreads library's PTHREAD_PROCESS_SHARED feature for condition and mutex variables and the mmap/munmap system calls.

You will implement two programs:

- **Host:** Initializes the shared data structures based on the number of philosophers (N), which is provided as a command line argument. After the initialization stage it creates N processes that execute the Philosopher program. It should pass the 2nd command line argument (M or the number of iterations) to these processes through exec system call.
- **Philosopher:** Implements a philosopher that sits in a loop that executes M times. M is provided as a command line argument. Inside the loop, each philosopher should print its state {THINKING, HUNGRY, and EATING} on the terminal as it goes through various stages. Your solution should use a barrier to make sure that all philosophers pass the barrier before they start their loops.

Important Note: It is strongly recommended that you check the semaphore solution for inter-process communication provided at https://computing.llnl.gov/tutorials/pthreads/man/pthread_mutexattr_init.txt You are free to use that semaphore solution to implement the synchronization data structures you will need in your solution to the Dining Philosophers problem.

Submission: You should submit all your source code along with a Readme file and preferably a Makefile on CANVAS.