AN OVERVIEW:

The 1st Question of the Assignment Contains solving various instances of Dining Philosopher's Problems. For the first 2 parts, the only restraining condition is the utilization of forks. For the next 2 parts, the Philosopher only has access to 2 bowls. For Each of these problems, 1 code uses mutex, a primary locking mechanism, whereas the other uses Semaphores for synchronization.

Detailed Explanation:

For the first part of the standard problem, I took the 5 mutex locks, each signifying a fork. Then when a fork was under use, its use was locked for the other philosophers. Only when the philosopher puts down the fork is the mutex lock unlocked and available for other philosophers. For the code using semaphores, the majority of the code stayed the same. The only difference was that instead of locking, sem_wait was used, and instead of unlocking, sem_post() was used.

For the Second Part of the problem, I took 5 mutex locks as forks and 2 mutex locks as bowls. Then when the philosophers were ready to eat, only when the bowl was free could the philosophers eat. For the code using semaphores, the same principles were applied. The only difference was that instead of locking, sem_wait was used, and instead of unlocking, sem_post() was used.