Harman Singh
Roll No. - 2019042
OS Assignment 4 WriteUp

-A counting semaphore is designed as a structure using the mutex pthread_mutex_t and a variable count is a field of the counting semaphore called my_semaphore.

-The wait() function takes a pointer to the counting semaphore and if the mutex of the structure is already blocked, the function returns. Else, the value of count variable is incremented and if the value becomes k-1, the mutex is blocked and no more philosophers can acquire the room.

-The signal function also takes a pointer to the counting sempahore and the count field is decremented inside the function. If the count becomes k-2 by some thread inside the function that solves the dining philosophers' problem(func), the mutex is unblocked and a philosopher can acquire the room.

-The variable k, array of forks represented by pthread_mutex_t named forks, pair of sauce bowls also represented as pthread_mutex_t named sauceBowls and the counting semaphore my_semaphore are all global variables.

-The maximum value of k is assumed to be 10000000.

-In the main function, all the mutexes(sauce bowls and forks) are initialised using pthread_mutex_init and the threads (representing philosophers) are created using pthread_create. The threads are then waited for completion and are joined using pthread_join. The mutexes are then destroyed after the completion of one cycle of the simulation.

-Inside the function pointer func which is used to solve the dining philosophers. The wait function is used to limit the number of philosophers in the room in order to prevent deadlock. The forks of the incoming philosopher are then locked. The pair of sauceBowls is also locked. Locking is done using pthread_mutex_lock API.

-The philosopher then starts eating. Once the philosopher has eaten, the forks and the sauce bowls are released using pthread_mutex_unlock and the the signal function is then used to reduce the amount of philosophers in the room.

-The blocking version is ph1_2019042.c and the non-blocking variant is ph2_2019042.c.

-In the non-blocking variant, pthread_mutex_trylock() API is used instead in the wait() function.

-The simulation then runs indefinitely..