Assignment 4

In this assignment I have tried to solve the modified dinning philosopher problem. I have created my own semaphore structure called my_semaphore which has value of semaphore and number of processes which are waiting to execute. I have implemented my own wait(), signal(), and sem_init() functions and using pthread_mutex library to achieve mutual exclusion through wait() and signal() functions.

In modified dinning philosopher I have solved the problem of bowls. I have given a room to philosophers for waiting until other philosophers are eating and a pair of bowls to any philosopher who is eating. I have printed all the process for the same. I have hardcoded the number of philosophes and it should be greater than equal to 2.

In blocking variant I am using chopsticks and bowls as binary semaphores and room as counting semaphore whereas in non-blocking, chopsticks as binary and room and bowls as counting semaphore. In wait function I am using pthread_mutex_lock to lock mutex resources and pthread_mutex_unlock to unlock them. I have used pthread_cond_wait() to unlock the mutex just before it sleeps, and pthread_cond_signal.
In non-blocking instead of using pthread_cond_t I am returning -1 from wait function when my value of semaphore is less than 0 and running it into an infinite loop until value of semaphore becomes non negative i.e. >=0.