Assignment 4
(By Vasudev Singhal Roll no 2019126)

In this assignment we have a dining philosophers problem with a pair of sauce bowls, both of which are needed simultaneously for a philosopher to eat.
To implement this we create a counting semaphore structure using mutex called my_semaphore so that multiple pthreads can synchronize their access to some limited/shared resources.

We had to make our own counting semaphores in order to solve the dining philosopher's problem in this assignment with additional 2 eating bowls.

We can see that at a time only one of the philosophers could be eating as there are only 2 bowls available and each philosopher requires 2 bowls for eating.

Firstly I initialize all the eating forks, philosophers and bowls using the functions initialize_sema, thread create, thread_join . Thread_create is used to create all the num philosophers by me.

In the Philosopher function , the left fork and right fork are assigned to each and every philosopher except the last(Assignment is done in the opposite manner for the last philosopher in order to prevent deadlock).Then, since each philosopher requires 2 eating bowls, I assign them.

In my wait function , firstly I lock the semaphore to block multiple accesses by various threads. Then I decrease the value of the semaphore to check the condition of the thread going to the critical section . After all the checking is done, I unlock the semaphore.

In my signal function , firstly I lock the semaphore to block multiple accesses by various threads. Then I increase the value of the semaphore and check the critical section before unlocking the semaphore again.

The pthread_mutex_lock function gets replaced with the pthread_mutex_trylock in both signal and wait functions for unlocking semaphore

Resources :-

https://medium.com/swlh/the-dining-philosophers-problem-solution-in-c-90e2593f64e8
https://sites.cs.ucsb.edu/~rich/class/cs170/notes/Semaphores/