**Ayush Mahant**
**2019353**
**section-a**

## Write up for assignment 4

So, the condition of the question is 2 sauce bowls and k forks and k philosophers, in a room, as according to the dining philosophers problem. Forks and pairs of bowls are shared by k philosophers. Now according to the question, race condition has to be avoided, so a semaphore named “my_semphore” is created.

The semaphore contains value or f, which states the availability of the threads. It makes sure that the thread which needs to acquire the waiting thread, gives up the lock and thus gets acquired by the requesting thread. The cont or the conditional variable, puts on condition, signal on the threads, which requests the signal and waits on semaphore. The mutex, ensures synchronization.

We make an array of forks semaphore and a bowl semaphore. To initialize and declare it, we call the initialize function. We have also allocated space to forks through malloc.

Then we create forks, and bowl for every philosopher through a while loop. Then, the wait function is used to wait on the bowl until it is free, and same for the two forks. It then decrements the struct attribute to check if the value is <0 . So waiting continues by cont'd wait, which ends when it is >1.

Then the mutex lock is realized to the critical section and same for the forks. Then a signal is sent which starts by acquiring mutex and thus the value is synchronized, then after decrementing it it checks for <0 and increases the value. Then thread signals to give up lock which is released and the critical section part is free. Then the no blocking part we just use the try wait until that part is free.

For the signal print value , we just print the availability of the semaphore.