Assignment 4: Write Up

- For the counting semaphore, a struct was used, which had the members as value(int) and lock(mutex).
- The wait functionality was implemented using the pthread_mutex_unlock and pthread_mutex_trylock functions.
- The signal functionality was implemented using the pthread_mutex_unlock and pthread_mutex_lock functions.
- For debugging purposes only, printSignal function was also implemented.
- The main problem was solved in funcPhilosopher function
- For handling the deadlock conditions, “pickFork” semaphore was used.
- It mandated that which of the philosopher was alloed to pick up the fork.
- Apart from this, to satisfy the condition that the philosopher also requires both the bowls, two more semaphores by the name of bowl1 and bowl2 were used.
- To make sure that the program runs flawlessly, a combination of signal and wait were used.

Output:

```
pw@ubuntu:~/Desktop/Sem3/OS/Assignment4$ make
gcc a4_2019440.c -lpthread -lrt -o a4
./a4
Enter the number of philsophers: 5
Philosopher 3 eats using fork number 3 and fork number 4.
Philosopher 4 eats using fork number 4 and fork number 0.
Philosopher 3 eats using fork number 3 and fork number 4.
Philosopher 2 eats using fork number 2 and fork number 3.
Philosopher 1 eats using fork number 1 and fork number 2.
Philosopher 0 eats using fork number 0 and fork number 1.
Philosopher 4 eats using fork number 4 and fork number 0.
Philosopher 3 eats using fork number 3 and fork number 4.
Philosopher 4 eats using fork number 4 and fork number 0.
Philosopher 0 eats using fork number 0 and fork number 1.
Philosopher 3 eats using fork number 3 and fork number 4.
```