Shubhan Singh
2022300118
Comps B

OS exp 7: Dining Philosophers problem

Solution: We made each thread (representing a philosopher) sleep for random amounts of time to simulate eating and thinking. To simulate the picking up of forks, we maintain state of each fork, whether its free or taken, then make any thread lock the chopstick semaphore as soon as it is free.
If both of the chopstics are not free, we just run an infinite while loop until one becomes free.
To simulate putting the chopstick down, we just free both the semaphores for any thread.

Learnings: We learnt how to write multithreaded programs using semaphores in this experiment. We learnt about the use of the <semaphore.h> library and its functions like sem_wait, sem_post, etc. Semaphores can be used to limit the number of process which can access the critical region simultaneously. We used binary semaphores (aka mutexes) in this program. We also learnt the syntax for the sem_init function, which lets us initialise a semaphore with a certain value (the max. number of process that can access the critical region).

Errors encountered: I encountered an error while trying to time all of the actions and realised that the clock() function maintains a seperate clock for each thread, so I used the time() function instead. Not many other errors were seen except for some syntax errors or silly mistakes in the code.