README

-Mayank Raj(B19CSE053)

Part 1:

Implementation details:

- Implemented deadlock-free version of dining philosopher problem using threads.
- Implemented deadlock version for dining philosopher problem using resource allocation graph.
- Cycles detection in the resource allocation graph is used for deadlock detection.
- In case of deadlock, we are freeing tha philosopher for that time.

How to run:

- Open a terminal and write
- gcc no_deadlock.c -lpthread;./a.out
- Open another terminal and write
- gcc with_deadlock.c -lpthread;./a.out

Results:

- Each philosopher's action is printed as they start thinking to, till they free the forks.
- In without deadlock solution, both the forks are taken simultaneously.
- In with deadlock solution, the philosopher takes the left fork first then the right fork.

Part2:

Solutions for starvation prevention are:

1. One of the methods is to limit the no. of readers to a particular threshold. If the threshold is k, if no. of readers is less than the threshold, then the writers are made to wait and if the threshold is exceeded and writers are waiting then the reader will unlock and the writer will proceed. This can be implemented using Semaphores.
2. Another method could be to assign severity of writing to the writer process each time when writer process comes. Whenever a writer's process comes, it will check the severity or urgency of writing, if greater than the particular threshold then the reader can unlock and the writer can start writing, or else the reader can continue.
3. Another solution can be to blend the above two approaches. We can have severity/urgency as well as a threshold on no. of readers. We can unlock the readers when their numbers are high or when a higher severity writer process comes in.