

## README - QUESTION 1

### Dining Philosophers Problem Implementation

This C code demonstrates the classic dining philosophers problem using pthreads for multithreading. The scenario involves a fixed number of philosophers seated around a table, alternating between thinking and eating with a shared set of forks.

#### Key Components:

- Philosopher Behavior: Each philosopher goes through a continuous cycle of thinking and attempting to eat.
- Resource Management: The philosophers acquire two forks (mutex locks) to eat, aiming to prevent conflicts and deadlocks.
- Concurrency Control: Mutex locks (`pthread_mutex_t`) manage access to forks and a shared bowl while a condition variable (`pthread_cond_t`) helps regulate the number of philosophers eating simultaneously.
- Deadlock Prevention: The code avoids deadlocks by enforcing a specific lock acquisition order for the forks, ensuring that no circular wait situation occurs.