FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

Parallel Programing – Assingment Report

Team Name - ATTİLA

Süleyman Burak GÜL - 190316034

Hamza KARAKAYA - 190316021

Dining Philosophers Problem Introduction

The provided Python code implements the solution to the classical dining philosophers problem using threads and synchronization primitives. Below is a detailed report explaining the code structure, the dining philosophers problem, and how the visualization works.

Dining Philosophers Problem:

The dining philosophers problem is a classic synchronization problem that illustrates the challenges of resource sharing and deadlock avoidance. The problem is defined as follows:

- There are N philosophers sitting around a circular dining table.
- Each philosopher alternates between thinking and eating.
- To eat, a philosopher must acquire two forks, one from their left and one from their right.
- The challenge is to design a solution that avoids deadlock and ensures that every philosopher gets a chance to eat.

Code Structure:

1. **Fork Class:**
    - Represents a fork on the dining table.
    - Each fork has an index, a lock for synchronization, and a queue for communication.
    - It can be picked up and put down by philosophers.
2. **Philosopher Class:**
    - Represents a philosopher as a thread.
    - Has methods for thinking, eating, and running the thread.
    - Uses left and right forks for eating.
    - Communicates actions (pickup, put down, eating) through a shared queue.
3. **Animated Table Function:**
    - Uses Matplotlib for visualizing the dining philosophers.
    - Updates the visualization based on philosopher actions.
    - Handles animation frames, positions of philosophers, and the state of forks.
4. **Table Function:**
    - Provides a simple text-based representation of the dining table.
    - Shows the state of each philosopher (eating or thinking) and the state of forks.

5. **Main Function:**
    - Initializes the number of philosophers (N) and spaghetti portions (M).
    - Creates forks and philosophers, ensuring each philosopher has a unique pair of forks.
    - Starts philosopher threads and a table thread for text-based visualization.
    - Calls the animated_table function for graphical visualization.

Visualization:

- The graphical visualization uses Matplotlib to create a circular dining table.
- Each philosopher is represented by a circle, and forks are represented by lines.
- The color of circles and lines changes based on the philosopher's state (thinking, eating) and fork state (picked up or not).

Conclusion:

The code successfully addresses the dining philosophers problem by ensuring proper synchronization using locks and a queue for communication. The visualization aids in understanding the dynamics of philosopher interactions and fork states during the dining process.