#### Dining Philosophers Problem

# Aim:

To implement a program for solving Dining Philosophers problem using Semaphores.

# Description:

The Dining Philosophers problem is a classic synchronization problem in computer science and concurrency theory. It illustrates the challenges of resource allocation and deadlock prevention in a concurrent system.

The philos() function represents the behavior of each philosopher. When a philosopher is thinking, they need to acquire both the left and right chopsticks before they can eat. This is done by waiting on the semaphores representing the chopsticks (sem\_wait).

After acquiring both chopsticks, the philosopher can eat for a period of time (in this case, simulated by sleep(2)).

After finishing eating, the philosopher releases both chopsticks (sem\_post), allowing other philosophers to use them.

Finally, the philosopher goes back to thinking.

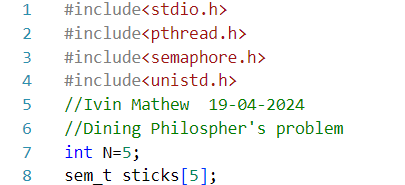
In the main function, the code initializes the semaphores for the chopsticks, setting their initial value to 1 (available).

Then, it creates 5 threads (one for each philosopher) using pthread\_create, passing the philosopher number (arg[i]) to the philos() function.

Finally, it waits for all philosopher threads to finish using pthread\_join.

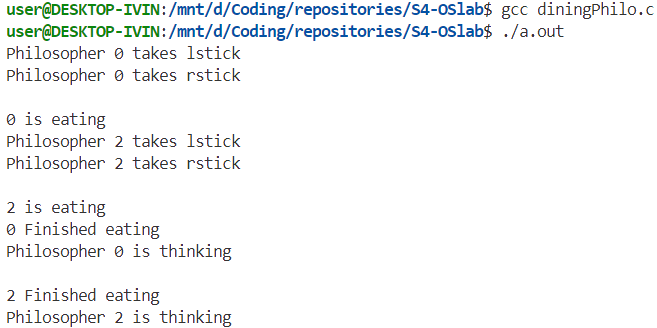
This ensures that no two neighboring philosophers can eat simultaneously, thus preventing deadlock

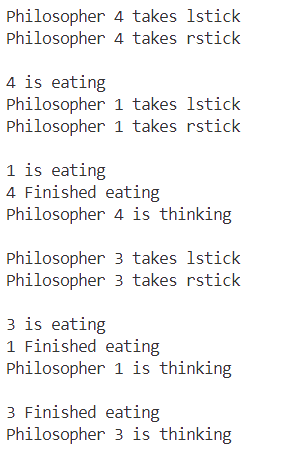
# Code



# 

# Sample Output





# Result

The program has been executed and output has been verified