**EXPERIENT 12**

**Design a C program to simulate the concept of Dining-Philosophers problem**

## AIM :

To design a C program to simulate the concept of Dining-Philosophers problem

## ALGORITHM :

* 1. Initialize Semaphores:
  + Create an array of semaphores, one for each philosopher, to control access to the chopsticks.
  + Initialize each semaphore to 1, indicating that the chopstick is available.
  1. Define Philosopher Structure:
  + Define a structure to represent a philosopher, which includes their ID and the semaphores representing the left and right chopsticks.
  1. Philosopher Lifecycle:
  + Each philosopher runs as a separate thread. In the thread function:
  + Think: Philosopher thinks for a random amount of time.
  + Pick up Chopsticks: Philosopher tries to pick up the left and right chopsticks (acquire semaphores). If both are available, the philosopher proceeds to eat. If not, they release the acquired chopstick(s) and return to thinking.
  + Eat: Philosopher eats for a random amount of time.
  + Put Down Chopsticks: Philosopher releases the left and right chopsticks (release semaphores) after eating.
  + Repeat the above steps indefinitely to simulate the philosopher's life cycle.
  1. Implement Deadlock Avoidance:
  + To avoid deadlock, impose a constraint such that a philosopher can only pick up both chopsticks if both are available.
  + One way to achieve this is by introducing a global semaphore that limits the number of philosophers allowed to pick up chopsticks simultaneously. For example, if there are 5 philosophers, allow at most 4 philosophers to pick up chopsticks simultaneously. This prevents the circular wait condition and avoids deadlock.
  1. Compile and Run:
  + Compile the program with appropriate multithreading flags (e.g., - pthread for gcc on Unix-based systems).
  + Run the program to observe the dining philosophers problem simulation.

## A screenshot of a computer Description automatically generatedOUTPUT :