Program-6

Aim: Write a program to simulate the concept of Dining-Philosopher's problem.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
sem_t room;              // Semaphore to control access to the dining room
sem_t chopstick[5];      // Semaphores for each chopstick at the table
// Function prototypes
void *philosopher(void *);
void eat(int);
int main() {
   int i, a[5];
   pthread_t tid[5];
   sem_init(&room, 0, 4);  // Initialize the room semaphore with a maximum count of 4
   for (i = 0; i < 5; i++)
      sem_init(&chopstick[i], 0, 1);  // Initialize each chopstick semaphore with a maximum count of 1
   for (i = 0; i < 5; i++) {
      a[i] = i;
      pthread_create(&tid[i], NULL, philosopher, (void *)&a[i]);  // Create 5 philosopher threads
   }
   for (i = 0; i < 5; i++)
      pthread_join(tid[i], NULL);  // Wait for all philosopher threads to finish
}
// Function for the philosopher thread
void *philosopher(void *num) {
   int phil = *(int *)num;
   sem_wait(&room);  // Wait for access to the dining room
   printf("\nPhilosopher %d has entered the room", phil);
   sem_wait(&chopstick[phil]);  // Wait for the left chopstick
```

```
    sem_wait(&chopstick[(phil + 1) % 5]);  // Wait for the right chopstick (circular arrangement)
    eat(phil);  // Call the eat function to simulate eating
    sleep(2);   // Simulate eating time
    printf("\nPhilosopher %d has finished eating", phil);
    sem_post(&chopstick[(phil + 1) % 5]);  // Release the right chopstick
    sem_post(&chopstick[phil]);  // Release the left chopstick
    sem_post(&room);  // Release the dining room for others
    // The philosopher goes back to thinking and potentially waiting for the room and chopsticks again
}
// Function to simulate the philosopher eating
void eat(int phil) {
    printf("\nPhilosopher %d is eating", phil);
}
```

Sample output:

```
Philosopher 0 has entered the room
Philosopher 0 is eating
Philosopher 1 has entered the room
Philosopher 1 is eating
Philosopher 2 has entered the room
Philosopher 2 is eating
Philosopher 3 has entered the room
Philosopher 3 is eating
Philosopher 4 is thinking
Philosopher 4 has entered the room
Philosopher 4 is eating
Philosopher 0 has finished eating
Philosopher 1 has finished eating
Philosopher 2 has finished eating
Philosopher 3 has finished eating
Philosopher 4 has finished eating
```