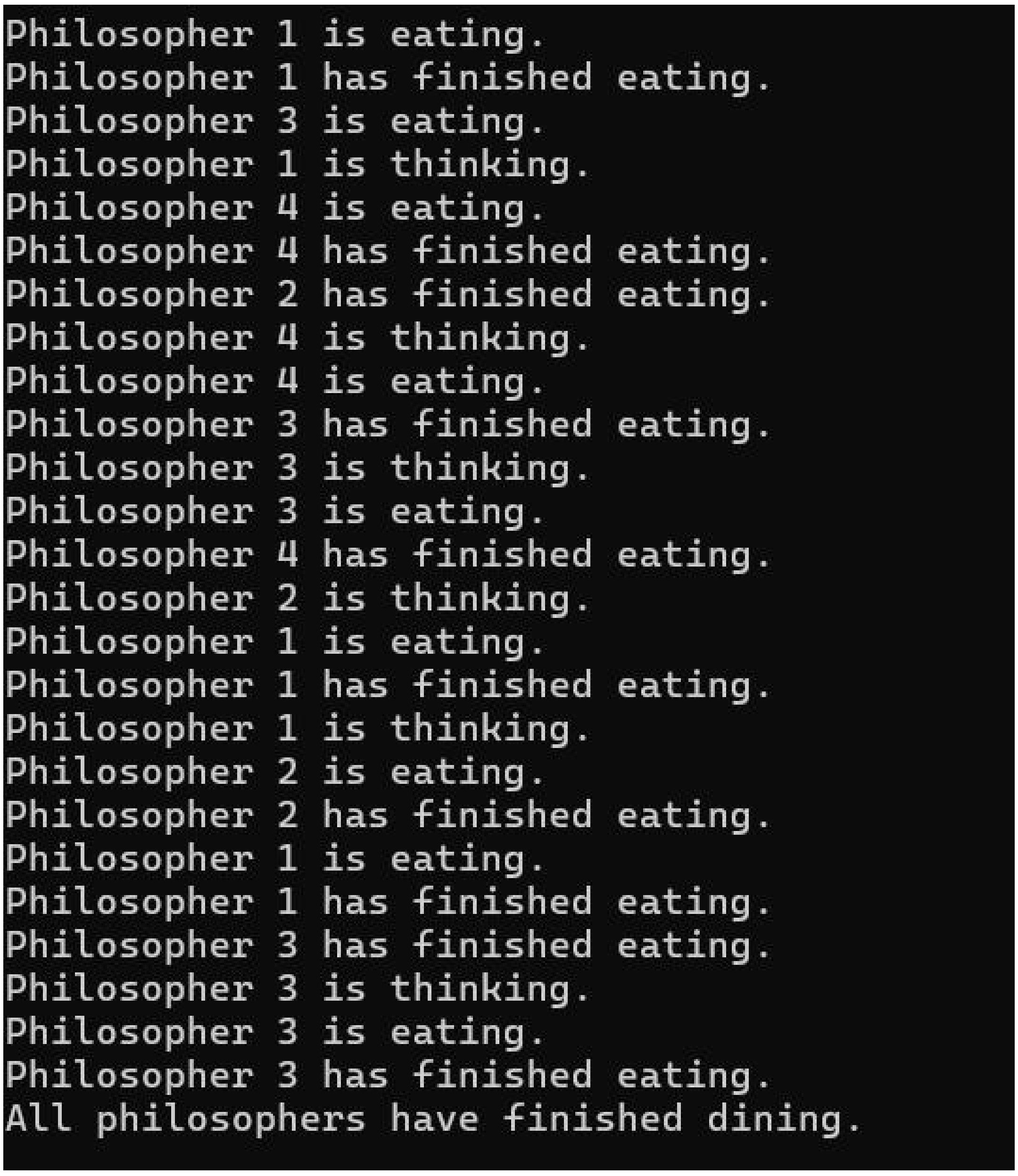
14. Design a C program to simulate the concept of Dining-Philosophers problem

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

#define NUM_PHILOSOPHERS 5

sem_t forks[NUM_PHILOSOPHERS];
pthread_t philosophers[NUM_PHILOSOPHERS];

void* philosopher(void* arg) {
   int id = *(int*)arg;
   int leftFork = id;
   int rightFork = (id + 1) % NUM_PHILOSOPHERS;

   for (int i = 0; i < 3; i++) {  // Simulating 3 meals
      printf("Philosopher %d is thinking.\n", id);

      // Pick up forks
      sem_wait(&forks[leftFork]);
      sem_wait(&forks[rightFork]);

      printf("Philosopher %d is eating.\n", id);

      // Put down forks
      sem_post(&forks[leftFork]);
      sem_post(&forks[rightFork]);

      printf("Philosopher %d has finished eating.\n", id);
   }

   free(arg);
   return NULL;
}

int main() {
   // Initialize semaphores
   for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
      sem_init(&forks[i], 0, 1);
   }
```

```
    // Create threads for philosophers
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        int* id = malloc(sizeof(int));
        *id = i;
        if (pthread_create(&philosophers[i], NULL, philosopher, id) != 0)
{
            perror("Failed to create philosopher thread");
            return 1;
        }
    }

    // Join philosopher threads
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_join(philosophers[i], NULL);
    }

    // Destroy semaphores
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        sem_destroy(&forks[i]);
    }

    printf("All philosophers have finished dining.\n");
    return 0;
}
```
Output:



```
Philosopher 1 is eating.
Philosopher 1 has finished eating.
Philosopher 3 is eating.
Philosopher 1 is thinking.
Philosopher 4 is eating.
Philosopher 4 has finished eating.
Philosopher 2 has finished eating.
Philosopher 4 is thinking.
Philosopher 4 is eating.
Philosopher 3 has finished eating.
Philosopher 3 is thinking.
Philosopher 3 is eating.
Philosopher 4 has finished eating.
Philosopher 2 is thinking.
Philosopher 1 is eating.
Philosopher 1 has finished eating.
Philosopher 1 is thinking.
Philosopher 2 is eating.
Philosopher 2 has finished eating.
Philosopher 1 is eating.
Philosopher 1 has finished eating.
Philosopher 3 has finished eating.
Philosopher 3 is thinking.
Philosopher 3 is eating.
Philosopher 3 has finished eating.
All philosophers have finished dining.
```