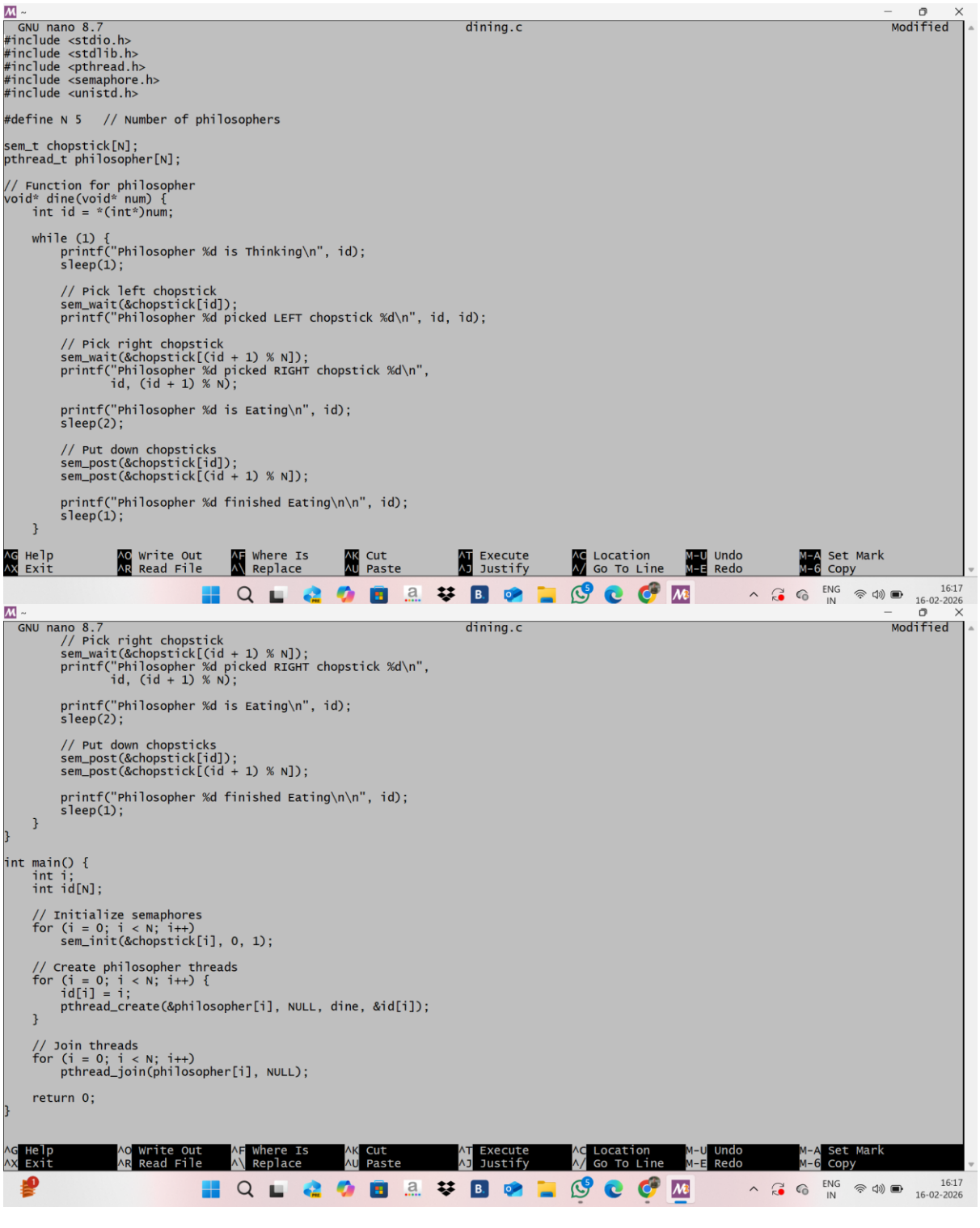
Considered there are N philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pick up the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both.

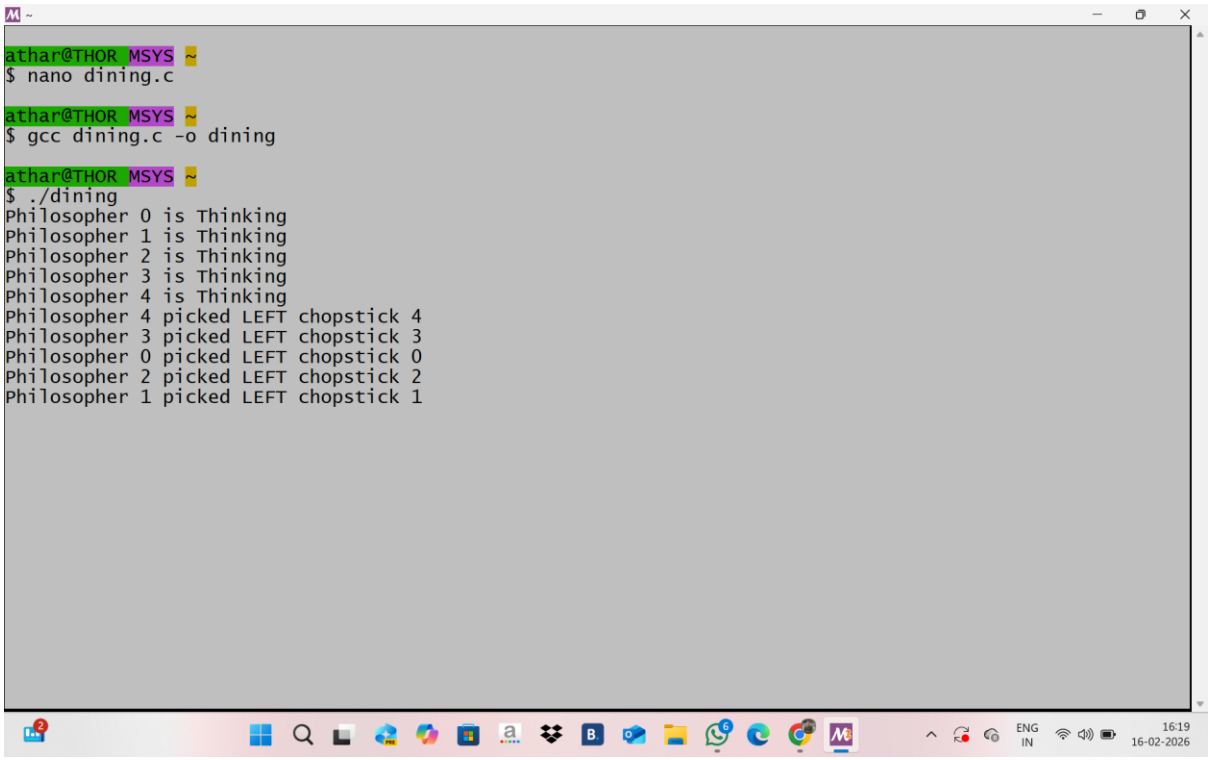
Write a program to solve the problem using process synchronization technique.



```
  GNU nano 8.7                                  dining.c                                  Modified
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define N 5   // Number of philosophers

sem_t chopstick[N];
pthread_t philosopher[N];

// Function for philosopher
void* dine(void* num) {
    int id = *(int*)num;

    while (1) {
        printf("Philosopher %d is Thinking\n", id);
        sleep(1);

        // Pick left chopstick
        sem_wait(&chopstick[id]);
        printf("Philosopher %d picked LEFT chopstick %d\n", id, id);

        // Pick right chopstick
        sem_wait(&chopstick[(id + 1) % N]);
        printf("Philosopher %d picked RIGHT chopstick %d\n",
               id, (id + 1) % N);

        printf("Philosopher %d is Eating\n", id);
        sleep(2);

        // Put down chopsticks
        sem_post(&chopstick[id]);
        sem_post(&chopstick[(id + 1) % N]);

        printf("Philosopher %d finished Eating\n\n", id);
        sleep(1);
    }

^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```



```
  GNU nano 8.7                                  dining.c                                  Modified
        // Pick right chopstick
        sem_wait(&chopstick[(id + 1) % N]);
        printf("Philosopher %d picked RIGHT chopstick %d\n",
               id, (id + 1) % N);

        printf("Philosopher %d is Eating\n", id);
        sleep(2);

        // Put down chopsticks
        sem_post(&chopstick[id]);
        sem_post(&chopstick[(id + 1) % N]);

        printf("Philosopher %d finished Eating\n\n", id);
        sleep(1);
    }
}

int main() {
    int i;
    int id[N];

    // Initialize semaphores
    for (i = 0; i < N; i++)
        sem_init(&chopstick[i], 0, 1);

    // Create philosopher threads
    for (i = 0; i < N; i++) {
        id[i] = i;
        pthread_create(&philosopher[i], NULL, dine, &id[i]);
    }

    // Join threads
    for (i = 0; i < N; i++)
        pthread_join(philosopher[i], NULL);

    return 0;
}

^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```


athar@THOR MSYS ~
$ nano dining.c
athar@THOR MSYS ~
$ gcc dining.c -o dining
athar@THOR MSYS ~
$ ./dining
Philosopher 0 is Thinking
Philosopher 1 is Thinking
Philosopher 2 is Thinking
Philosopher 3 is Thinking
Philosopher 4 is Thinking
Philosopher 4 picked LEFT chopstick 4
Philosopher 3 picked LEFT chopstick 3
Philosopher 0 picked LEFT chopstick 0
Philosopher 2 picked LEFT chopstick 2
Philosopher 1 picked LEFT chopstick 1
ENG
IN
16:19
16-02-2026