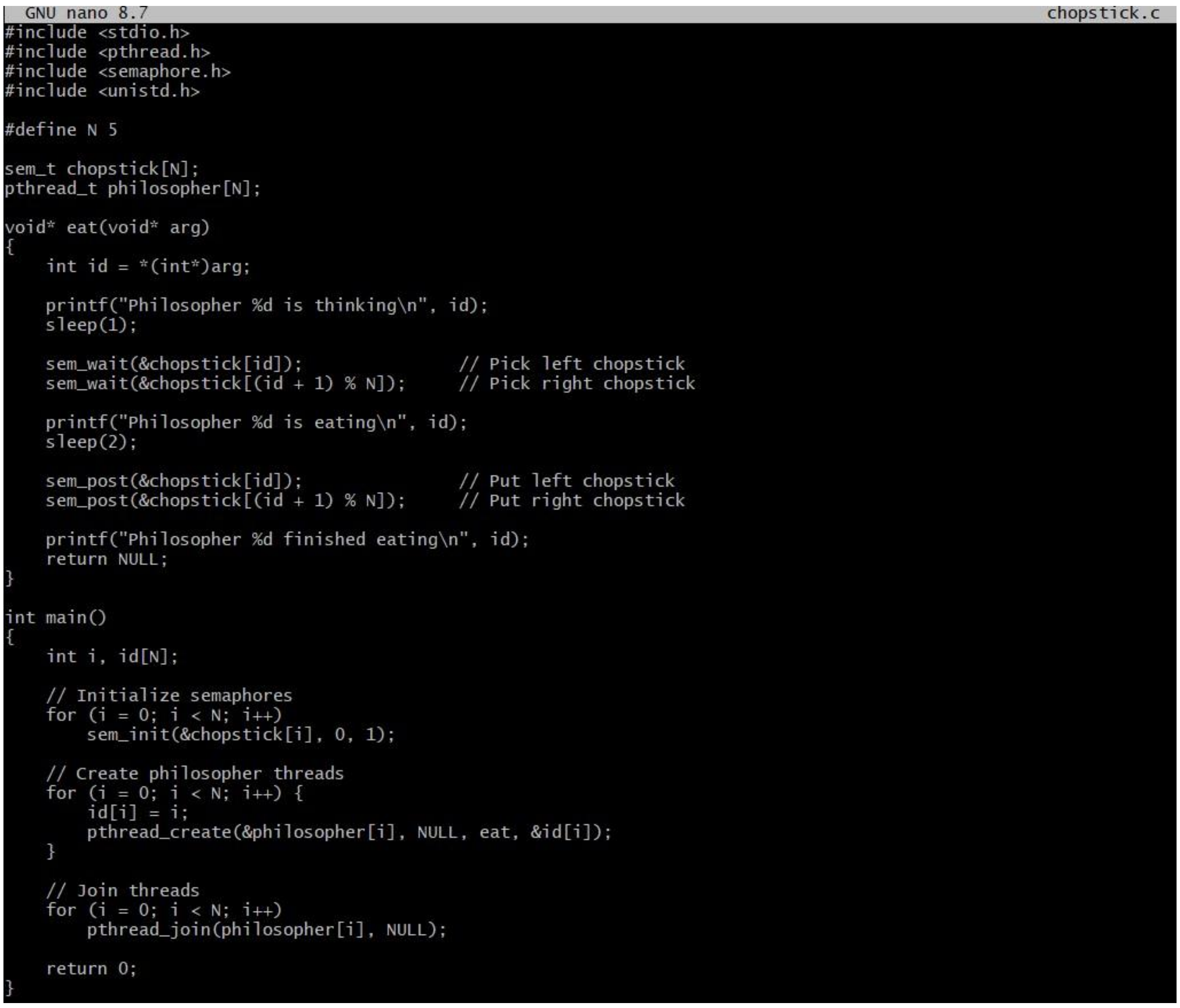
PracticalNO : 6

AIM :

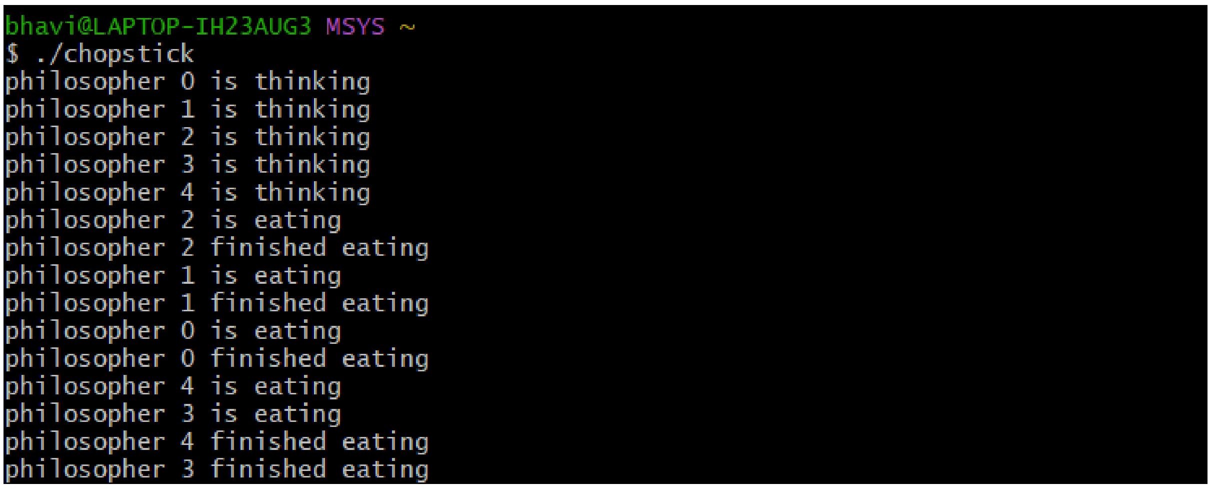
Considered there are N philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pick up the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent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

CODE :



```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define N 5

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

void* eat(void* arg)
{
    int id = *(int*)arg;

    printf("Philosopher %d is thinking\n", id);
    sleep(1);

    sem_wait(&chopstick[id]);               // Pick left chopstick
    sem_wait(&chopstick[(id + 1) % N]);     // Pick right chopstick

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

    sem_post(&chopstick[id]);               // Put left chopstick
    sem_post(&chopstick[(id + 1) % N]);     // Put right chopstick

    printf("Philosopher %d finished eating\n", id);
    return NULL;
}

int main()
{
    int i, 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, eat, &id[i]);
    }

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

    return 0;
}
```

OUTPUT:



```
bhavi@LAPTOP-IH23AUG3 MSYS ~
$ ./chopstick
philosopher 0 is thinking
philosopher 1 is thinking
philosopher 2 is thinking
philosopher 3 is thinking
philosopher 4 is thinking
philosopher 2 is eating
philosopher 2 finished eating
philosopher 1 is eating
philosopher 1 finished eating
philosopher 0 is eating
philosopher 0 finished eating
philosopher 4 is eating
philosopher 3 is eating
philosopher 4 finished eating
philosopher 3 finished eating
```