Practical 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:

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

#define N 5
#define THINKING 0
#define HUNGRY 1
#define EATING 2

int state[N];
int phil[N] = {0, 1, 2, 3, 4};

sem_t mutex;
sem_t S[N];

#define LEFT (phnum + N - 1) % N
#define RIGHT (phnum + 1) % N
```

Practical 6

```
void test(int phnum)
{
   if (state[phnum] == HUNGRY &&
      state[LEFT] != EATING &&
      state[RIGHT] != EATING)
   {
      state[phnum] = EATING;
      sleep(1);
      printf("Philosopher %d takes chopsticks %d and %d\n",
            phnum + 1, LEFT + 1, phnum + 1);
      printf("Philosopher %d is Eating\n", phnum + 1);
      sem_post(&S[phnum]);
   }
}

void take_chopstick(int phnum)
{
   sem_wait(&mutex);
   state[phnum] = HUNGRY;
   printf("Philosopher %d is Hungry\n", phnum + 1);
   test(phnum);
   sem_post(&mutex);
   sem_wait(&S[phnum]);
}

void put_chopstick(int phnum)
{
```

```
    sem_wait(&mutex);
    state[phnum] = THINKING;
    printf("Philosopher %d puts down chopsticks %d and %d\n",
          phnum + 1, LEFT + 1, phnum + 1);
    printf("Philosopher %d is Thinking\n", phnum + 1);
    test(LEFT);
    test(RIGHT);
    sem_post(&mutex);
}

void* philosopher(void* num)
{
    int* i = num;
    while (1)
    {
        sleep(1);
        take_chopstick(*i);
        sleep(1);
        put_chopstick(*i);
    }
}

int main()
{
    pthread_t thread_id[N];
    sem_init(&mutex, 0, 1);
```

Practical 6

```
    for (int i = 0; i < N; i++)
        sem_init(&S[i], 0, 0);

    for (int i = 0; i < N; i++)
    {
        pthread_create(&thread_id[i], NULL, philosopher, &phil[i]);
        printf("Philosopher %d is Thinking\n", i + 1);
    }

    for (int i = 0; i < N; i++)
        pthread_join(thread_id[i], NULL);

    return 0;
}
```

Output: