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

#define N 5
#define LEFT (i + N - 1) % N
#define RIGHT (i + 1) % N
#define THINKING 0
#define HUNGRY 1
#define EATING 2

int state[N]; pthread_t t[N]; sem_t s[N]; sem_t mutex;

void THINK(int n)
{  printf("The Philosopher %d is thinking \n", n);
   sleep(1);  }

void EAT(int n)
{  printf("Philosopher %d is eating\n", n);
   sleep(1);
   printf("Philosopher %d finished eating\n", n);    }

void TAKEFORK(int i)
{  sem_wait(&mutex);
   state[i] = HUNGRY;
   if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING)
   {  state[i] = EATING;
      sem_post(&s[i]);  }
   sem_post(&mutex);
   sem_wait(&s[i]);          }

void PUTFORKS(int i)
{  sem_wait(&mutex);
   state[i] = THINKING;
   if (state[LEFT] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING)

   {  state[LEFT] = EATING;
      sem_post(&s[LEFT]);   }
   if (state[RIGHT] == HUNGRY && state[LEFT] != EATING && state[RIGHT] !=
EATING)
   {  state[RIGHT] = EATING;
      sem_post(&s[RIGHT]);     }
   sem_post(&mutex);                }
```

```
void *philo(void *n)
{  int philo_id = *(int *)n;
   while (1)
   {  THINK(philo_id);
      TAKEFORK(philo_id);
      if (state[philo_id] == EATING)
      {  EAT(philo_id);    }
      PUTFORKS(philo_id);        }
   return NULL;                  }

void main()
{  int i;
   for (i = 0; i < N; i++)
   {   sem_init(&s[i], 0, 0);      }
   sem_init(&mutex, 0, 1);
   for (i = 0; i < N; i++)
   {  int *arg = malloc(sizeof(*arg));
      if (arg == NULL)
      {  perror("Unable to allocate memory for thread argument.");
         exit(EXIT_FAILURE);    }
      *arg = i;
      pthread_create(&t[i], NULL, philo, arg);      }
   for (i = 0; i < N; i++)
   {  pthread_join(t[i], NULL);     }                                    }
```