```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#define NUM_PHILOSOPHERS 5
#define NUM_CHOPSTICKS 5

void dine(int n);
pthread_t philosopher[NUM_PHILOSOPHERS];
pthread_mutex_t chopstick[NUM_CHOPSTICKS];

int main()
{
  int i, status_message;
  void *msg;
  for (i = 1; i <= NUM_CHOPSTICKS; i++)
  {
    status_message = pthread_mutex_init(&chopstick[i], NULL);
    if (status_message == -1)
    {
      printf("\n Mutex initialization failed");
      exit(1);
    }
  }
  for (i = 1; i <= NUM_PHILOSOPHERS; i++)
  {
    status_message = pthread_create(&philosopher[i], NULL, (void *)dine, (int *)i);
    if (status_message != 0)
    {
      printf("\n Thread creation error \n");
      exit(1);
    }
  }
  for (i = 1; i <= NUM_PHILOSOPHERS; i++)
  {
    status_message = pthread_join(philosopher[i], &msg);
    if (status_message != 0)
    {
      printf("\n Thread join failed \n");
      exit(1);
    }
  }
  for (i = 1; i <= NUM_CHOPSTICKS; i++)
  {
    status_message = pthread_mutex_destroy(&chopstick[i]);
    if (status_message != 0)
    {
      printf("\n Mutex Destroyed \n");
      exit(1);
    }
  }
  return 0;
}
void dine(int n)
{
```

```
  printf("\nPhilosopher % d is thinking ", n);
  pthread_mutex_lock(&chopstick[n]);
  pthread_mutex_lock(&chopstick[(n + 1) % NUM_CHOPSTICKS]);
  printf("\nPhilosopher % d is eating ", n);
  sleep(3);
  pthread_mutex_unlock(&chopstick[n]);
  pthread_mutex_unlock(&chopstick[(n + 1) % NUM_CHOPSTICKS]);
  printf("\nPhilosopher % d Finished eating ", n);
}
```

Output:

```
Philosopher  2 is thinking
Philosopher  2 is eating
Philosopher  3 is thinking
Philosopher  5 is thinking
Philosopher  5 is eating
Philosopher  1 is thinking
Philosopher  4 is thinking
Philosopher  4 is eating
Philosopher  2 Finished eating
Philosopher  5 Finished eating
Philosopher  1 is eating
Philosopher  4 Finished eating
Philosopher  3 is eating
Philosopher  1 Finished eating
Philosopher  3 Finished eating
```