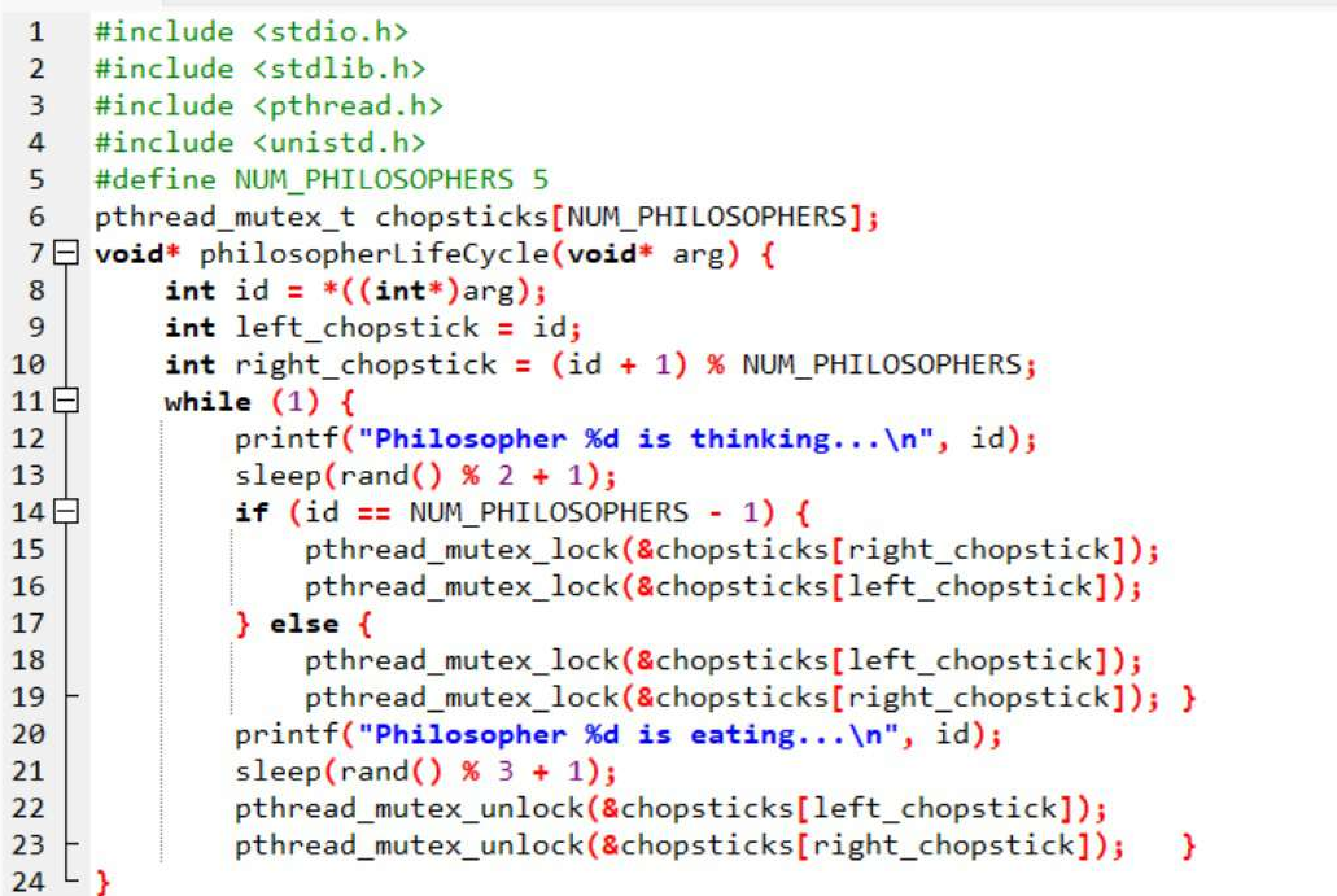
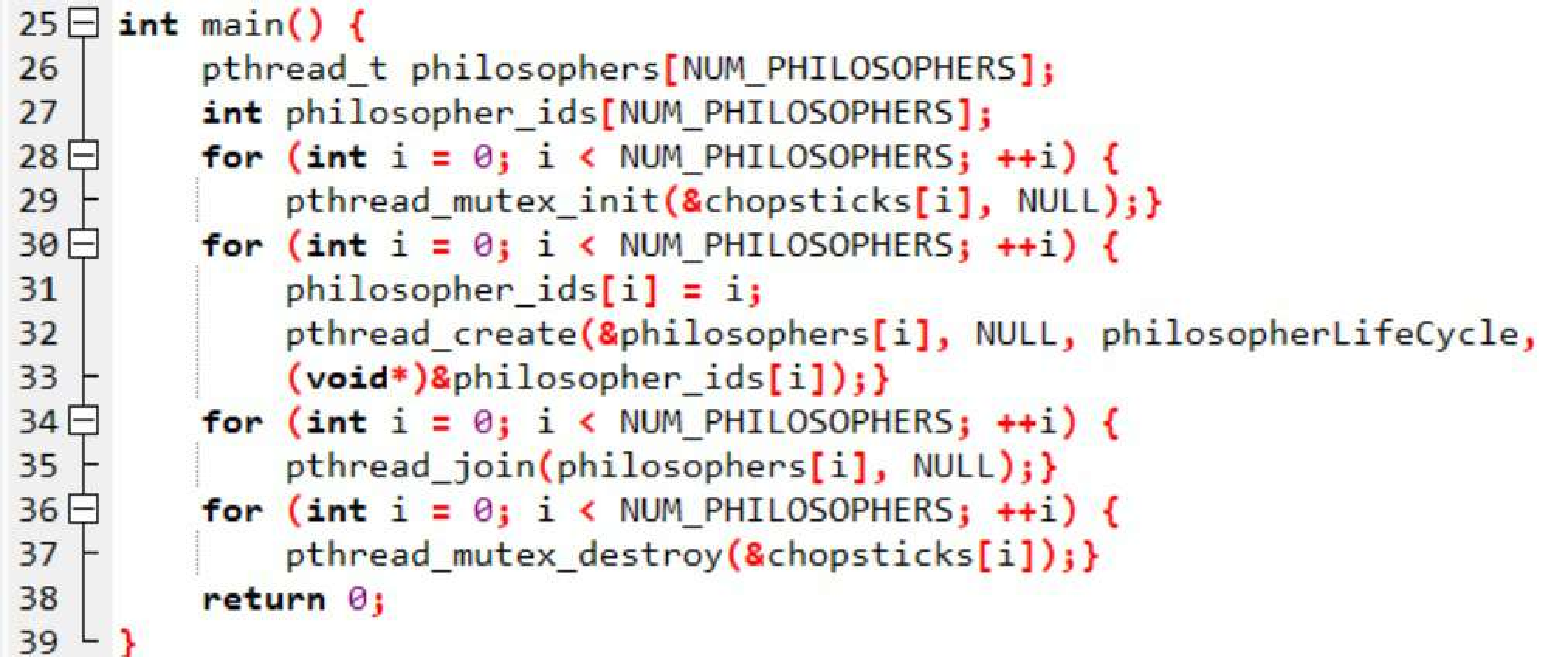
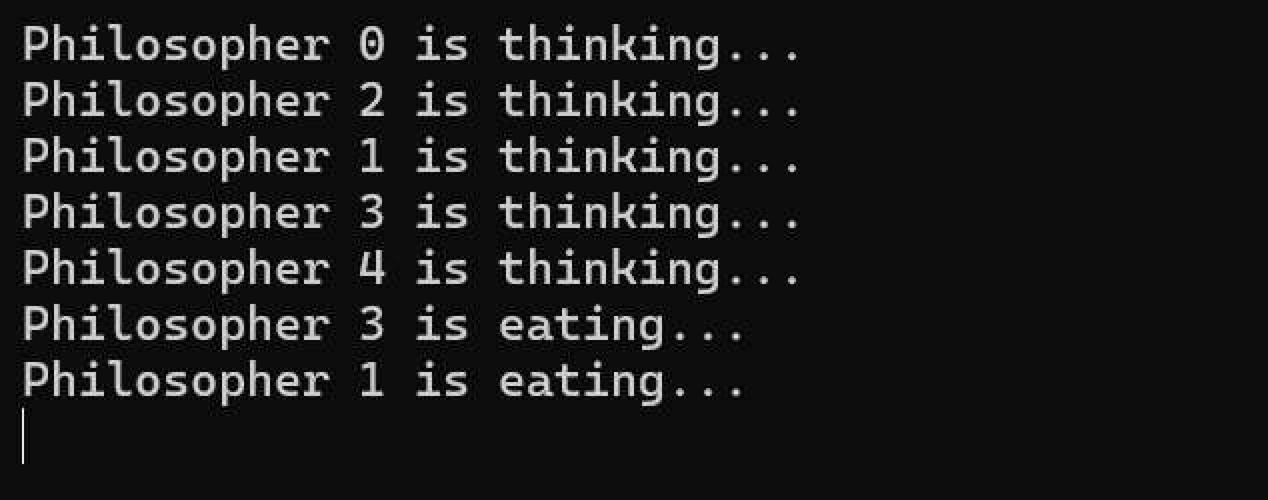


```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_PHILOSOPHERS 5
pthread_mutex_t chopsticks[NUM_PHILOSOPHERS];
void* philosopherLifeCycle(void* arg) {
    int id = *((int*)arg);
    int left_chopstick = id;
    int right_chopstick = (id + 1) % NUM_PHILOSOPHERS;
    while (1) {
        printf("Philosopher %d is thinking...\n", id);
        sleep(rand() % 2 + 1);
        if (id == NUM_PHILOSOPHERS - 1) {
            pthread_mutex_lock(&chopsticks[right_chopstick]);
            pthread_mutex_lock(&chopsticks[left_chopstick]);
        } else {
            pthread_mutex_lock(&chopsticks[left_chopstick]);
            pthread_mutex_lock(&chopsticks[right_chopstick]); }
        printf("Philosopher %d is eating...\n", id);
        sleep(rand() % 3 + 1);
        pthread_mutex_unlock(&chopsticks[left_chopstick]);
        pthread_mutex_unlock(&chopsticks[right_chopstick]);   }
}
```



```c
int main() {
    pthread_t philosophers[NUM_PHILOSOPHERS];
    int philosopher_ids[NUM_PHILOSOPHERS];
    for (int i = 0; i < NUM_PHILOSOPHERS; ++i) {
        pthread_mutex_init(&chopsticks[i], NULL);}
    for (int i = 0; i < NUM_PHILOSOPHERS; ++i) {
        philosopher_ids[i] = i;
        pthread_create(&philosophers[i], NULL, philosopherLifeCycle,
        (void*)&philosopher_ids[i]);}
    for (int i = 0; i < NUM_PHILOSOPHERS; ++i) {
        pthread_join(philosophers[i], NULL);}
    for (int i = 0; i < NUM_PHILOSOPHERS; ++i) {
        pthread_mutex_destroy(&chopsticks[i]);}
    return 0;
}
```



```
Philosopher 0 is thinking...
Philosopher 2 is thinking...
Philosopher 1 is thinking...
Philosopher 3 is thinking...
Philosopher 4 is thinking...
Philosopher 3 is eating...
Philosopher 1 is eating...
```