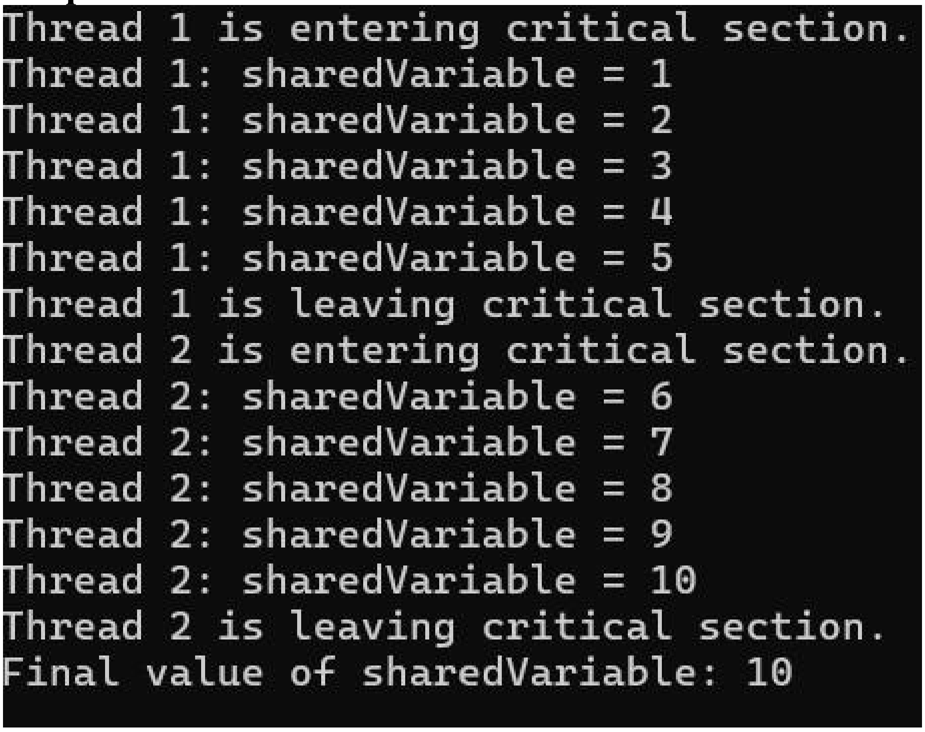
17. Design a C program to implement process synchronization using mutex locks.

Program:

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

pthread_mutex_t lock;
int sharedVariable = 0;

void* incrementFunction(void* arg) {
   long thread_id = *(long*)arg;  // Dereference and read the thread
ID
   pthread_mutex_lock(&lock);    // Acquire the lock
   printf("Thread %ld is entering critical section.\n", thread_id);

   for (int i = 0; i < 5; i++) {
      sharedVariable++;
      printf("Thread %ld: sharedVariable = %d\n", thread_id,
sharedVariable);
      sleep(1); // Simulate some work
   }

   printf("Thread %ld is leaving critical section.\n", thread_id);
   pthread_mutex_unlock(&lock); // Release the lock
   return NULL;
}

int main() {
   pthread_t thread1, thread2;
   long thread1_id = 1, thread2_id = 2;

   pthread_mutex_init(&lock, NULL); // Initialize mutex lock

   // Pass the addresses of the thread IDs
   pthread_create(&thread1, NULL, incrementFunction,
&thread1_id);
   pthread_create(&thread2, NULL, incrementFunction,
&thread2_id);

   // Wait for threads to finish
   pthread_join(thread1, NULL);
   pthread_join(thread2, NULL);
```

```
    pthread_mutex_destroy(&lock); // Destroy the mutex lock

    printf("Final value of sharedVariable: %d\n", sharedVariable);
    return 0;
}
```

Output:



```
Thread 1 is entering critical section.
Thread 1: sharedVariable = 1
Thread 1: sharedVariable = 2
Thread 1: sharedVariable = 3
Thread 1: sharedVariable = 4
Thread 1: sharedVariable = 5
Thread 1 is leaving critical section.
Thread 2 is entering critical section.
Thread 2: sharedVariable = 6
Thread 2: sharedVariable = 7
Thread 2: sharedVariable = 8
Thread 2: sharedVariable = 9
Thread 2: sharedVariable = 10
Thread 2 is leaving critical section.
Final value of sharedVariable: 10
```