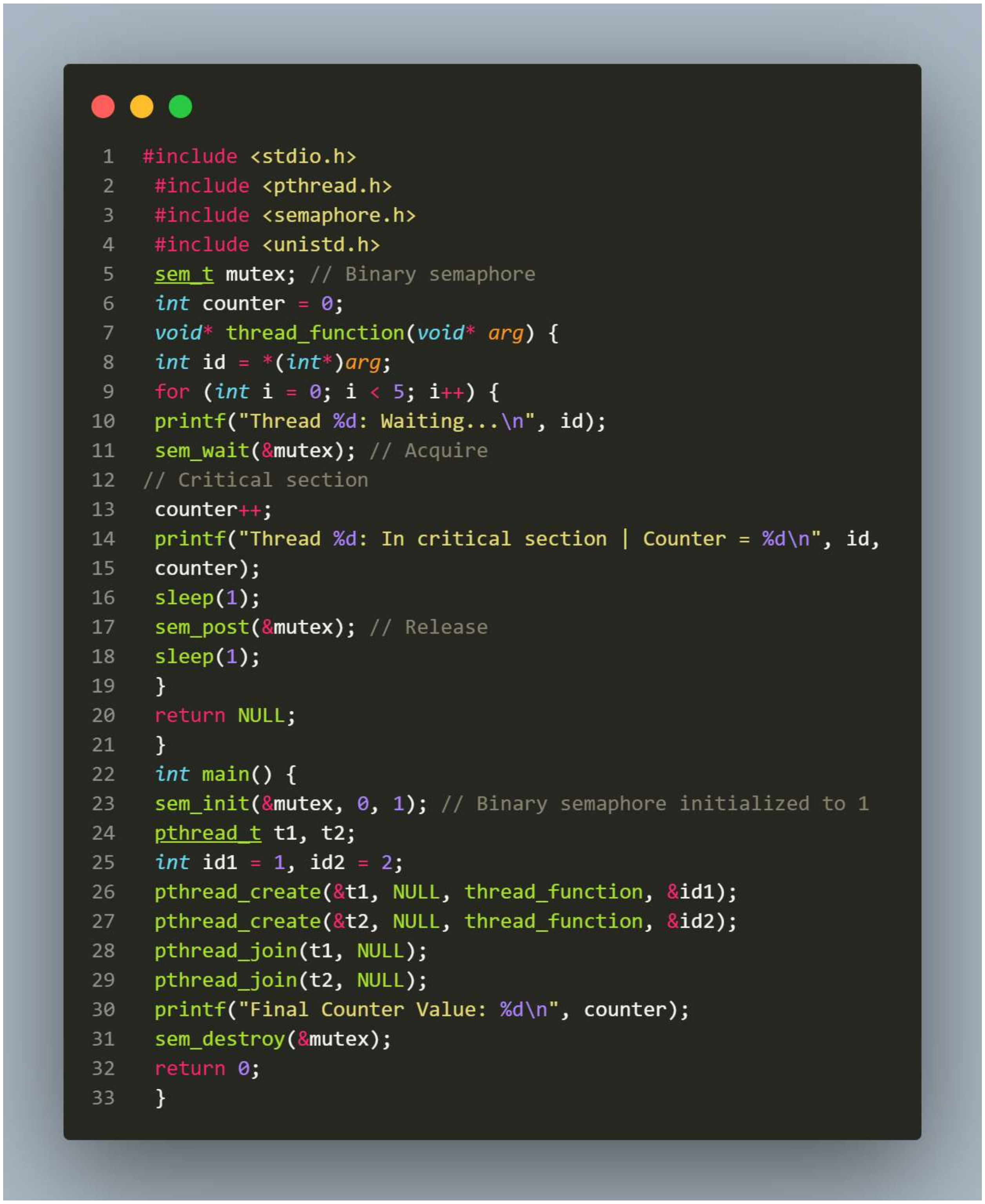
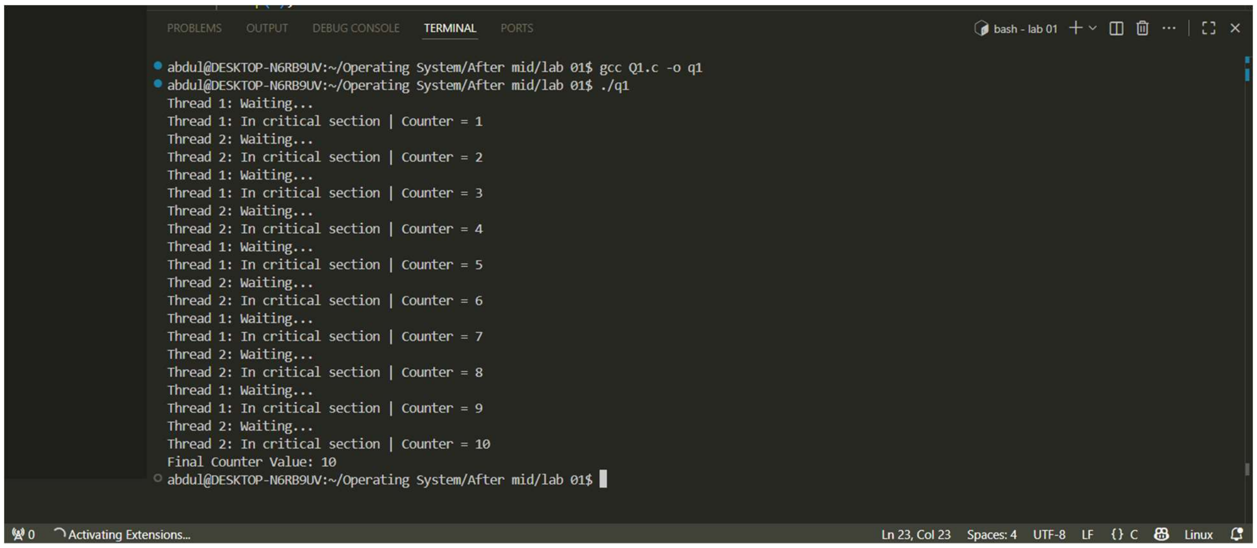


```c
#include <stdio.h>
 #include <pthread.h>
 #include <semaphore.h>
 #include <unistd.h>
 sem_t mutex; // Binary semaphore
 int counter = 0;
 void* thread_function(void* arg) {
 int id = *(int*)arg;
 for (int i = 0; i < 5; i++) {
 printf("Thread %d: Waiting...\n", id);
 sem_wait(&mutex); // Acquire
// Critical section
 counter++;
 printf("Thread %d: In critical section | Counter = %d\n", id,
 counter);
 sleep(1);
 sem_post(&mutex); // Release
 sleep(1);
 }
 return NULL;
 }
 int main() {
 sem_init(&mutex, 0, 1); // Binary semaphore initialized to 1
 pthread_t t1, t2;
 int id1 = 1, id2 = 2;
 pthread_create(&t1, NULL, thread_function, &id1);
 pthread_create(&t2, NULL, thread_function, &id2);
 pthread_join(t1, NULL);
 pthread_join(t2, NULL);
 printf("Final Counter Value: %d\n", counter);
 sem_destroy(&mutex);
 return 0;
 }
```

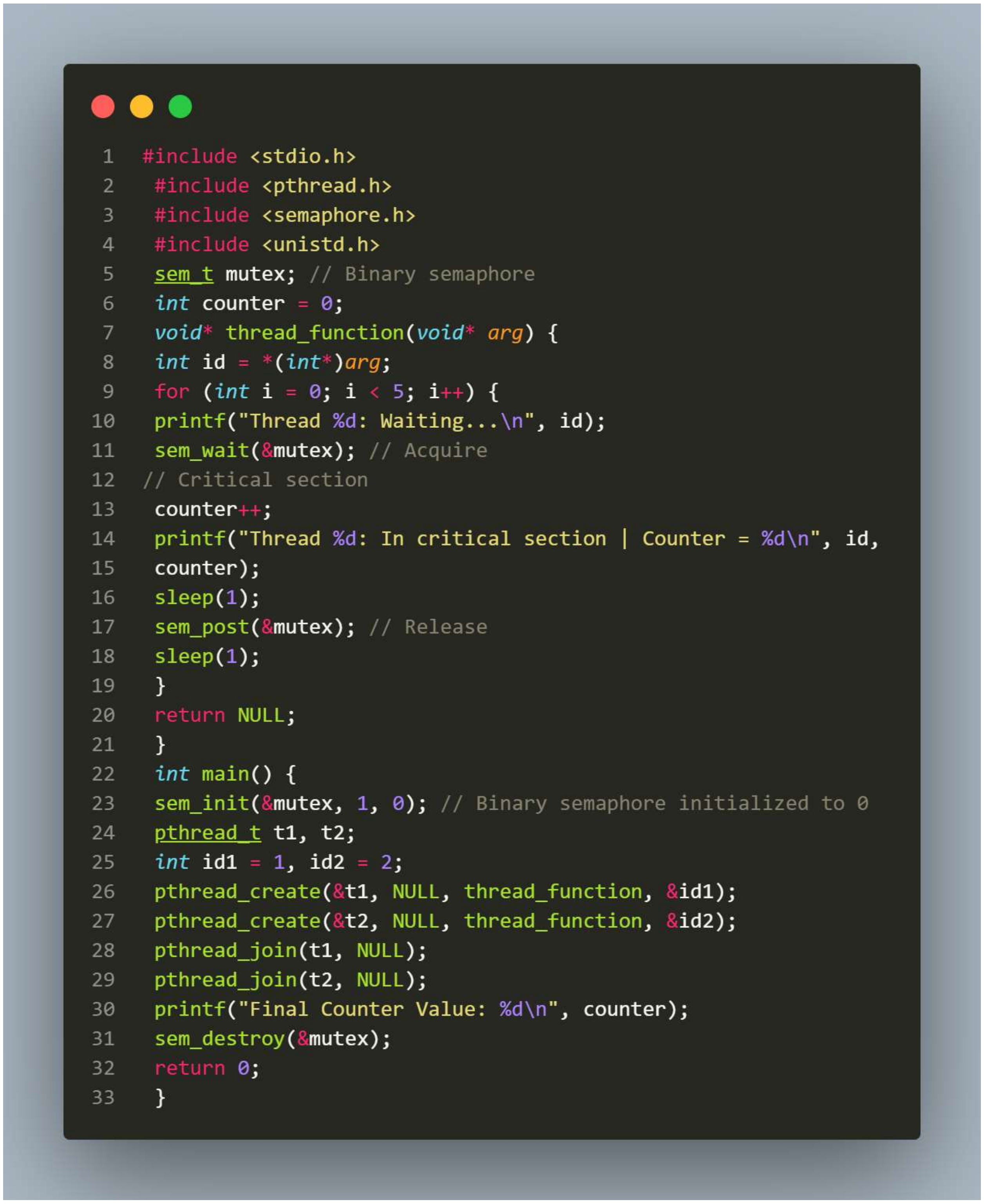
Output:



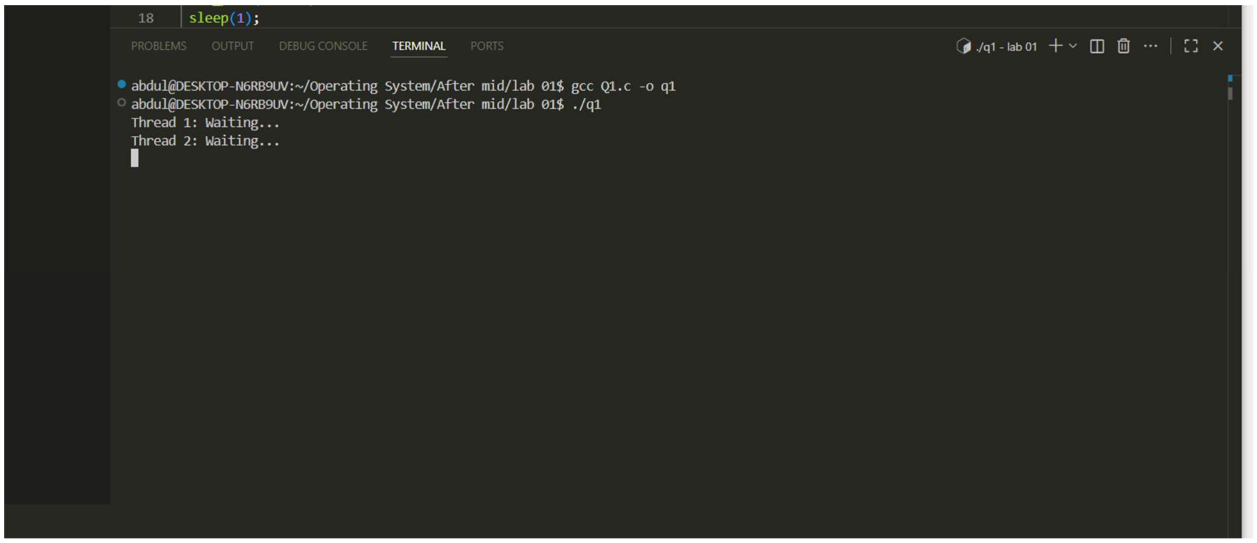
```
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ gcc Q1.c -o q1
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q1
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 2
Thread 1: Waiting...
Thread 1: In critical section | Counter = 3
Thread 2: Waiting...
Thread 2: In critical section | Counter = 4
Thread 1: Waiting...
Thread 1: In critical section | Counter = 5
Thread 2: Waiting...
Thread 2: In critical section | Counter = 6
Thread 1: Waiting...
Thread 1: In critical section | Counter = 7
Thread 2: Waiting...
Thread 2: In critical section | Counter = 8
Thread 1: Waiting...
Thread 1: In critical section | Counter = 9
Thread 2: Waiting...
Thread 2: In critical section | Counter = 10
Final Counter Value: 10
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$
```

DESCRIPTION:

1,0



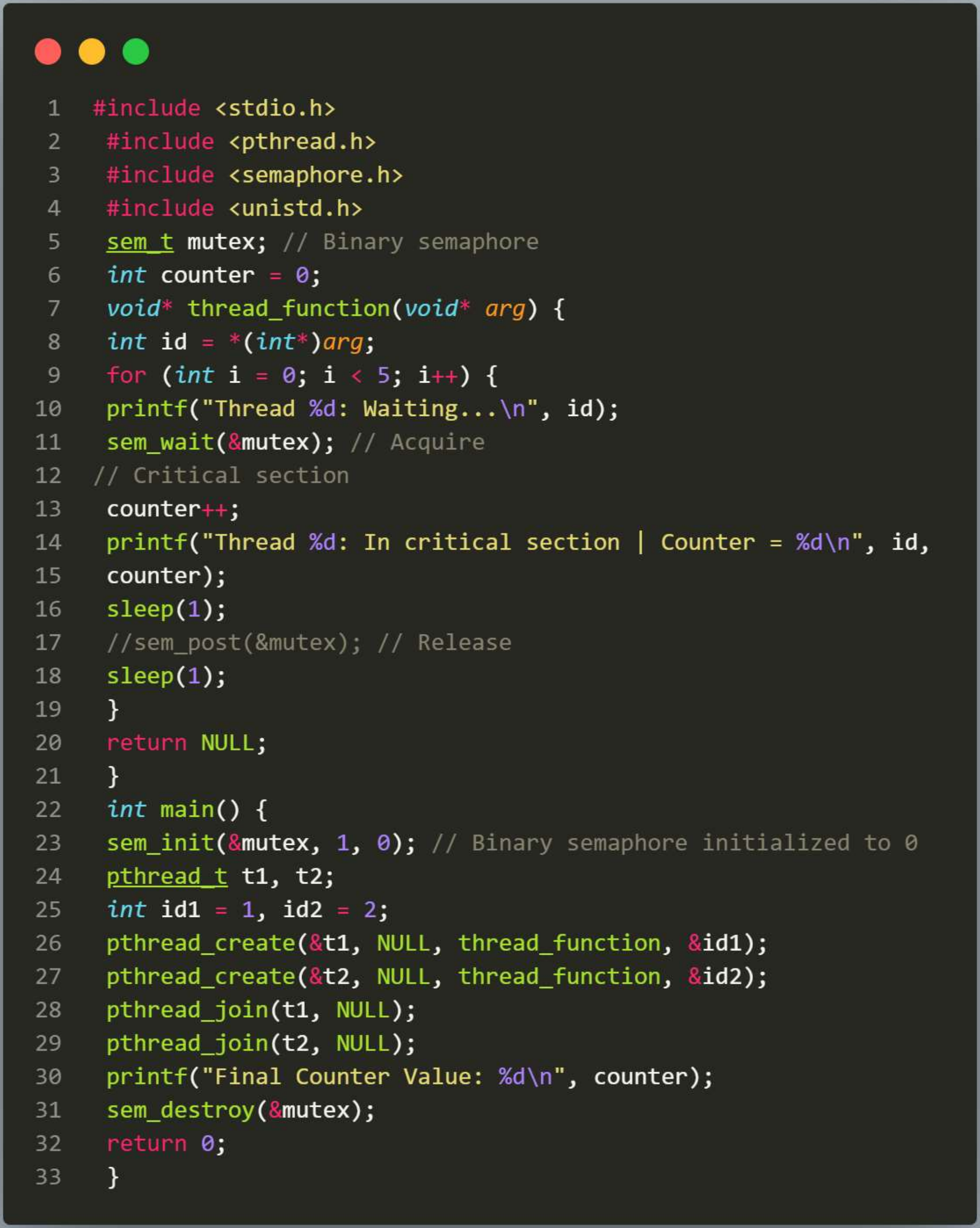
```c
#include <stdio.h>
 #include <pthread.h>
 #include <semaphore.h>
 #include <unistd.h>
 sem_t mutex; // Binary semaphore
 int counter = 0;
 void* thread_function(void* arg) {
 int id = *(int*)arg;
 for (int i = 0; i < 5; i++) {
 printf("Thread %d: Waiting...\n", id);
 sem_wait(&mutex); // Acquire
// Critical section
 counter++;
 printf("Thread %d: In critical section | Counter = %d\n", id,
 counter);
 sleep(1);
 sem_post(&mutex); // Release
 sleep(1);
 }
 return NULL;
 }
 int main() {
 sem_init(&mutex, 1, 0); // Binary semaphore initialized to 0
 pthread_t t1, t2;
 int id1 = 1, id2 = 2;
 pthread_create(&t1, NULL, thread_function, &id1);
 pthread_create(&t2, NULL, thread_function, &id2);
 pthread_join(t1, NULL);
 pthread_join(t2, NULL);
 printf("Final Counter Value: %d\n", counter);
 sem_destroy(&mutex);
 return 0;
 }
```



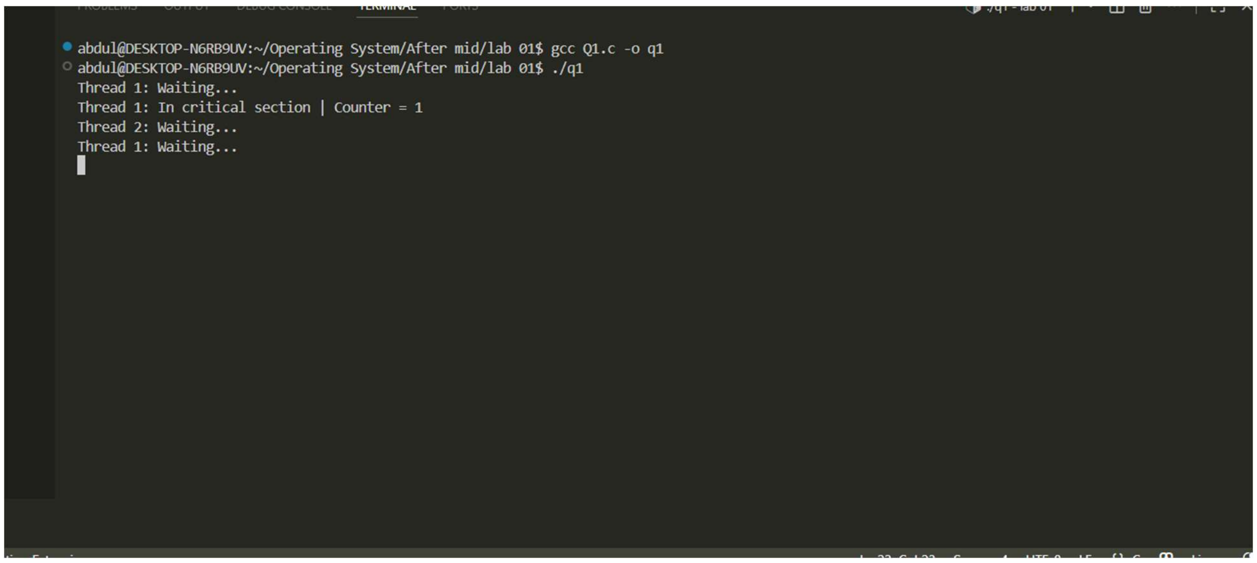
```
sleep(1);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

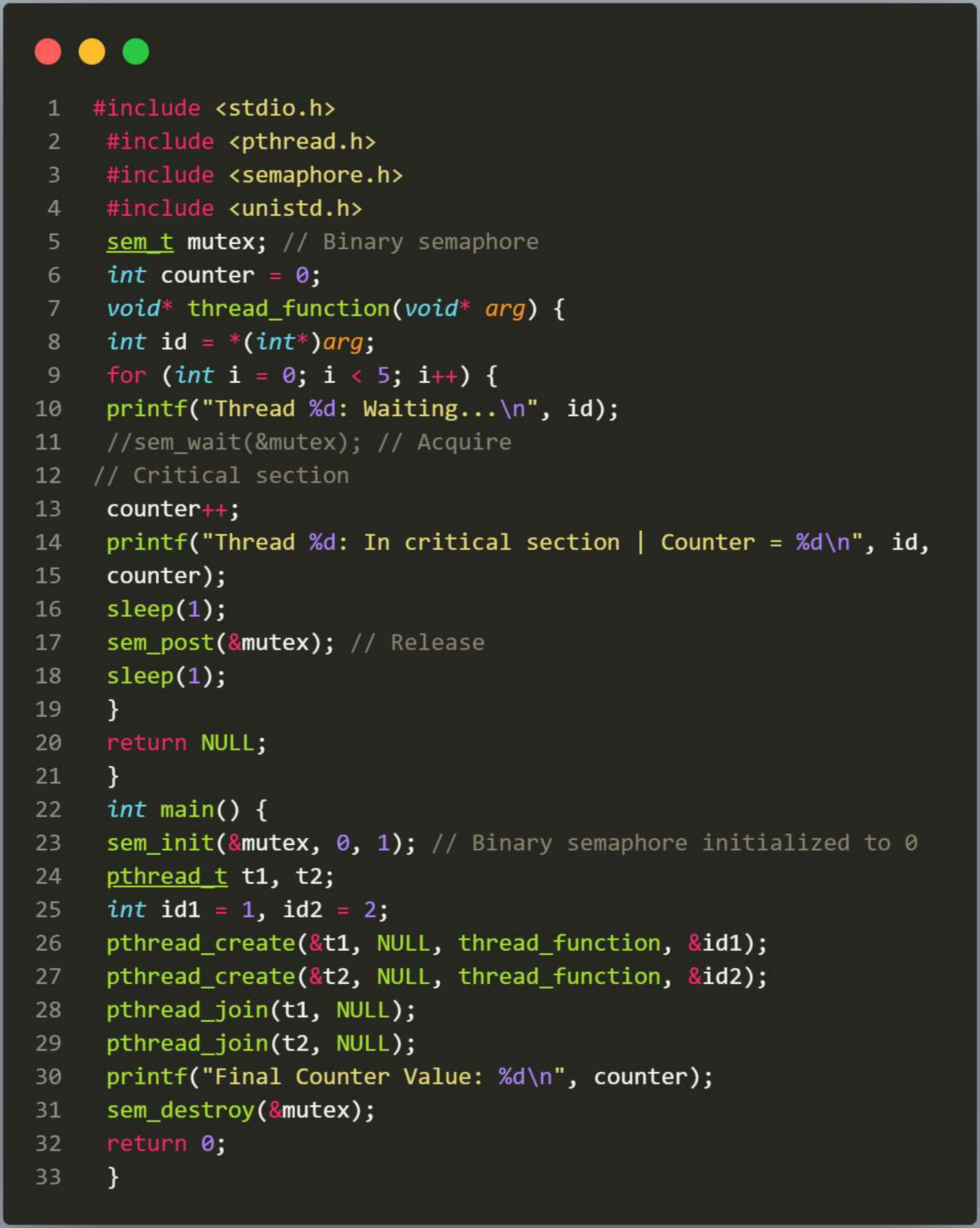
```
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ gcc Q1.c -o q1
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q1
Thread 1: Waiting...
Thread 2: Waiting...
```



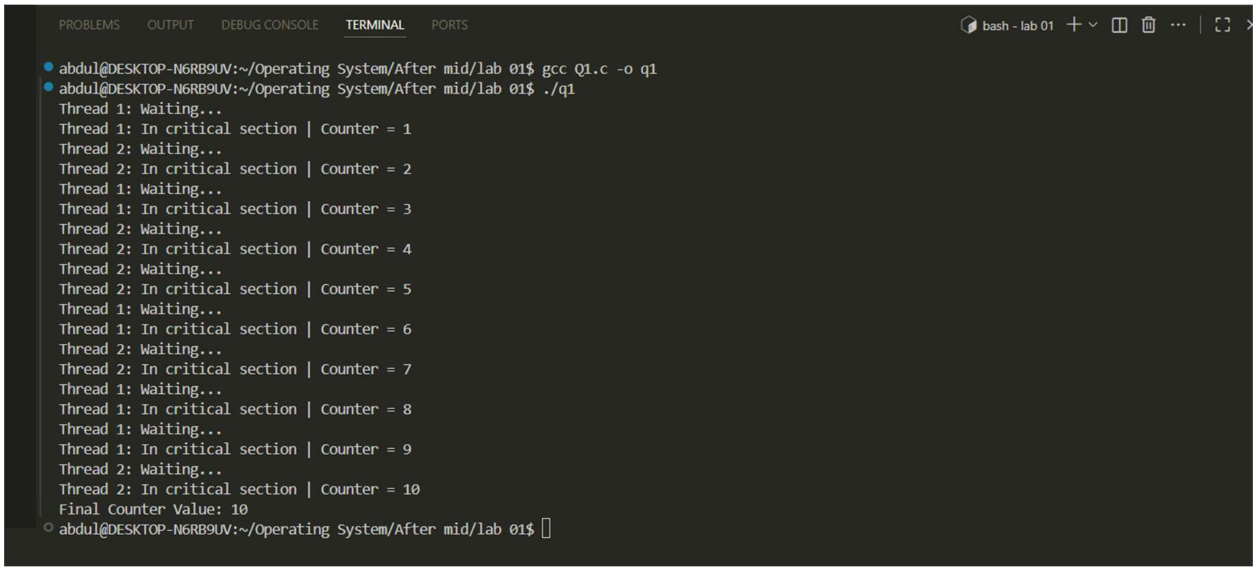
```c
#include <stdio.h>
 #include <pthread.h>
 #include <semaphore.h>
 #include <unistd.h>
 sem_t mutex; // Binary semaphore
 int counter = 0;
 void* thread_function(void* arg) {
 int id = *(int*)arg;
 for (int i = 0; i < 5; i++) {
 printf("Thread %d: Waiting...\n", id);
 sem_wait(&mutex); // Acquire
// Critical section
 counter++;
 printf("Thread %d: In critical section | Counter = %d\n", id,
 counter);
 sleep(1);
 //sem_post(&mutex); // Release
 sleep(1);
 }
 return NULL;
 }
 int main() {
 sem_init(&mutex, 1, 0); // Binary semaphore initialized to 0
 pthread_t t1, t2;
 int id1 = 1, id2 = 2;
 pthread_create(&t1, NULL, thread_function, &id1);
 pthread_create(&t2, NULL, thread_function, &id2);
 pthread_join(t1, NULL);
 pthread_join(t2, NULL);
 printf("Final Counter Value: %d\n", counter);
 sem_destroy(&mutex);
 return 0;
 }
```



```
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ gcc Q1.c -o q1
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q1
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 1: Waiting...
```

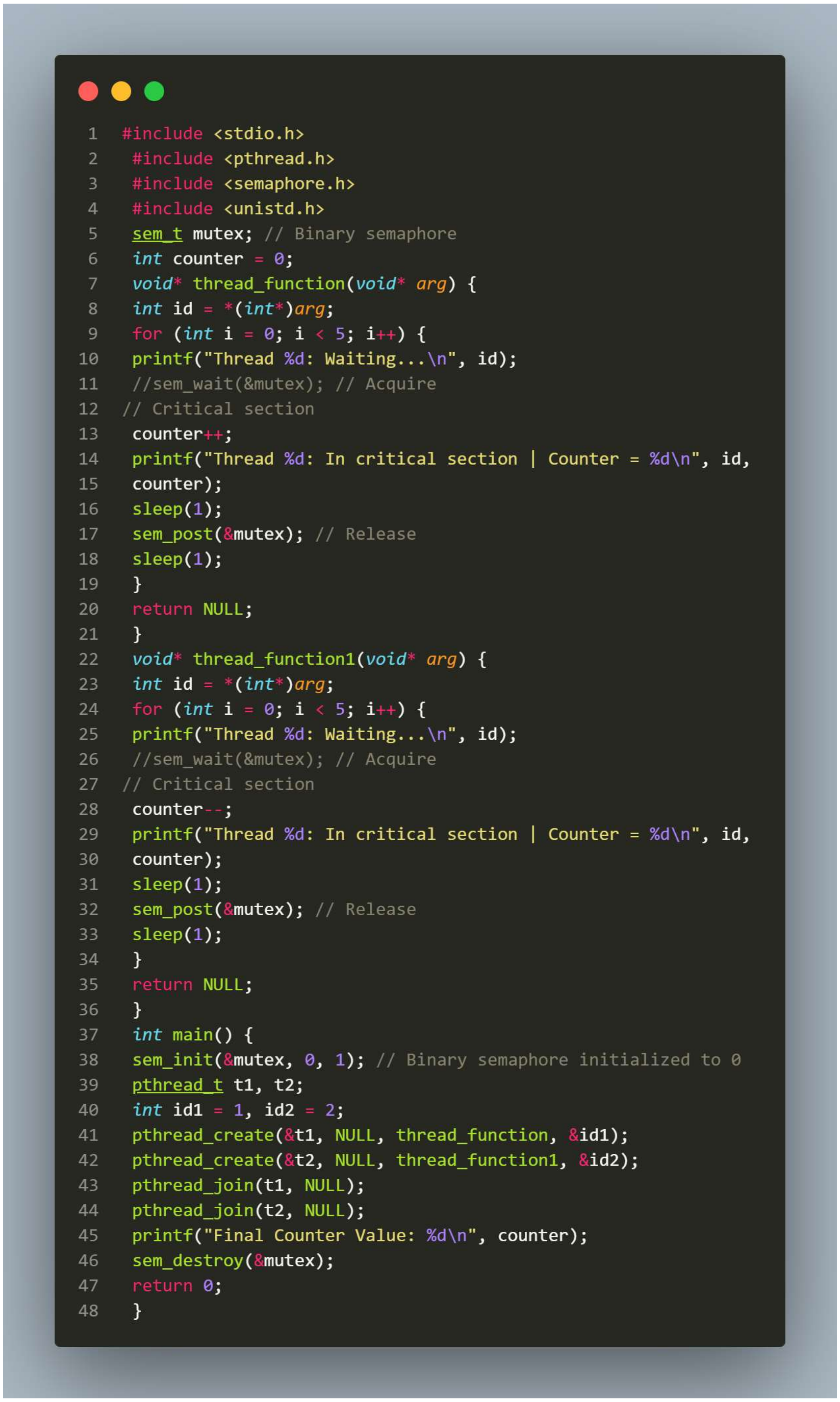


```c
#include <stdio.h>
 #include <pthread.h>
 #include <semaphore.h>
 #include <unistd.h>
 sem_t mutex; // Binary semaphore
 int counter = 0;
 void* thread_function(void* arg) {
 int id = *(int*)arg;
 for (int i = 0; i < 5; i++) {
 printf("Thread %d: Waiting...\n", id);
 //sem_wait(&mutex); // Acquire
// Critical section
 counter++;
 printf("Thread %d: In critical section | Counter = %d\n", id,
 counter);
 sleep(1);
 sem_post(&mutex); // Release
 sleep(1);
 }
 return NULL;
 }
 int main() {
 sem_init(&mutex, 0, 1); // Binary semaphore initialized to 0
 pthread_t t1, t2;
 int id1 = 1, id2 = 2;
 pthread_create(&t1, NULL, thread_function, &id1);
 pthread_create(&t2, NULL, thread_function, &id2);
 pthread_join(t1, NULL);
 pthread_join(t2, NULL);
 printf("Final Counter Value: %d\n", counter);
 sem_destroy(&mutex);
 return 0;
 }
```

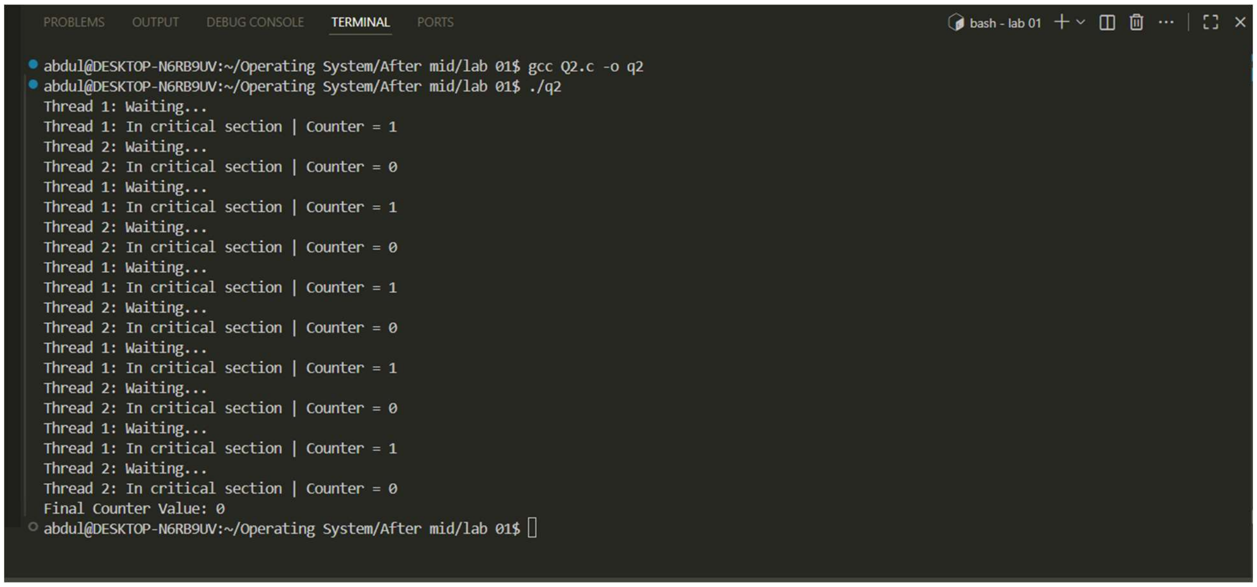

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ gcc Q1.c -o q1
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q1
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 2
Thread 1: Waiting...
Thread 1: In critical section | Counter = 3
Thread 2: Waiting...
Thread 2: In critical section | Counter = 4
Thread 2: Waiting...
Thread 2: In critical section | Counter = 5
Thread 1: Waiting...
Thread 1: In critical section | Counter = 6
Thread 2: Waiting...
Thread 2: In critical section | Counter = 7
Thread 1: Waiting...
Thread 1: In critical section | Counter = 8
Thread 1: Waiting...
Thread 1: In critical section | Counter = 9
Thread 2: Waiting...
Thread 2: In critical section | Counter = 10
Final Counter Value: 10
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$
```

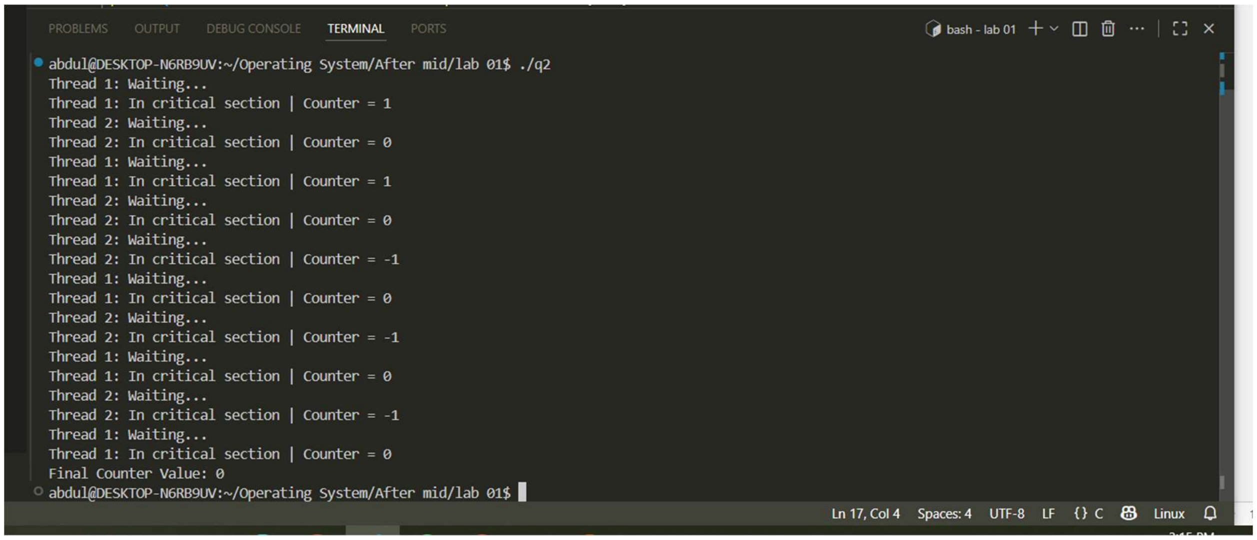
Task2:



```c
#include <stdio.h>
 #include <pthread.h>
 #include <semaphore.h>
 #include <unistd.h>
 sem_t mutex; // Binary semaphore
 int counter = 0;
 void* thread_function(void* arg) {
 int id = *(int*)arg;
 for (int i = 0; i < 5; i++) {
 printf("Thread %d: Waiting...\n", id);
 //sem_wait(&mutex); // Acquire
// Critical section
 counter++;
 printf("Thread %d: In critical section | Counter = %d\n", id,
 counter);
 sleep(1);
 sem_post(&mutex); // Release
 sleep(1);
 }
 return NULL;
 }
 void* thread_function1(void* arg) {
 int id = *(int*)arg;
 for (int i = 0; i < 5; i++) {
 printf("Thread %d: Waiting...\n", id);
 //sem_wait(&mutex); // Acquire
// Critical section
 counter--;
 printf("Thread %d: In critical section | Counter = %d\n", id,
 counter);
 sleep(1);
 sem_post(&mutex); // Release
 sleep(1);
 }
 return NULL;
 }
 int main() {
 sem_init(&mutex, 0, 1); // Binary semaphore initialized to 0
 pthread_t t1, t2;
 int id1 = 1, id2 = 2;
 pthread_create(&t1, NULL, thread_function, &id1);
 pthread_create(&t2, NULL, thread_function1, &id2);
 pthread_join(t1, NULL);
 pthread_join(t2, NULL);
 printf("Final Counter Value: %d\n", counter);
 sem_destroy(&mutex);
 return 0;
 }
```



```
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ gcc Q2.c -o q2
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q2
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Final Counter Value: 0
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$
```



```
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q2
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 2: Waiting...
Thread 2: In critical section | Counter = -1
Thread 1: Waiting...
Thread 1: In critical section | Counter = 0
Thread 2: Waiting...
Thread 2: In critical section | Counter = -1
Thread 1: Waiting...
Thread 1: In critical section | Counter = 0
Thread 2: Waiting...
Thread 2: In critical section | Counter = -1
Thread 1: Waiting...
Thread 1: In critical section | Counter = 0
Final Counter Value: 0
abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$
```

Mutex	**Semaphore**
In mutex one thread can access the resource	In semaphore multiple threads can access the resources based on the count
Works like a **lock**	Works like a **counter**
has only **two states**: locked/unlocked	has a **value** that will be 0 and 1

Mutex	**Semaphore**
Used for **mutual exclusion**	Used for **controlling access** to limited resources
Only the **owner** can release it	**Anyone** can signal (increase) it