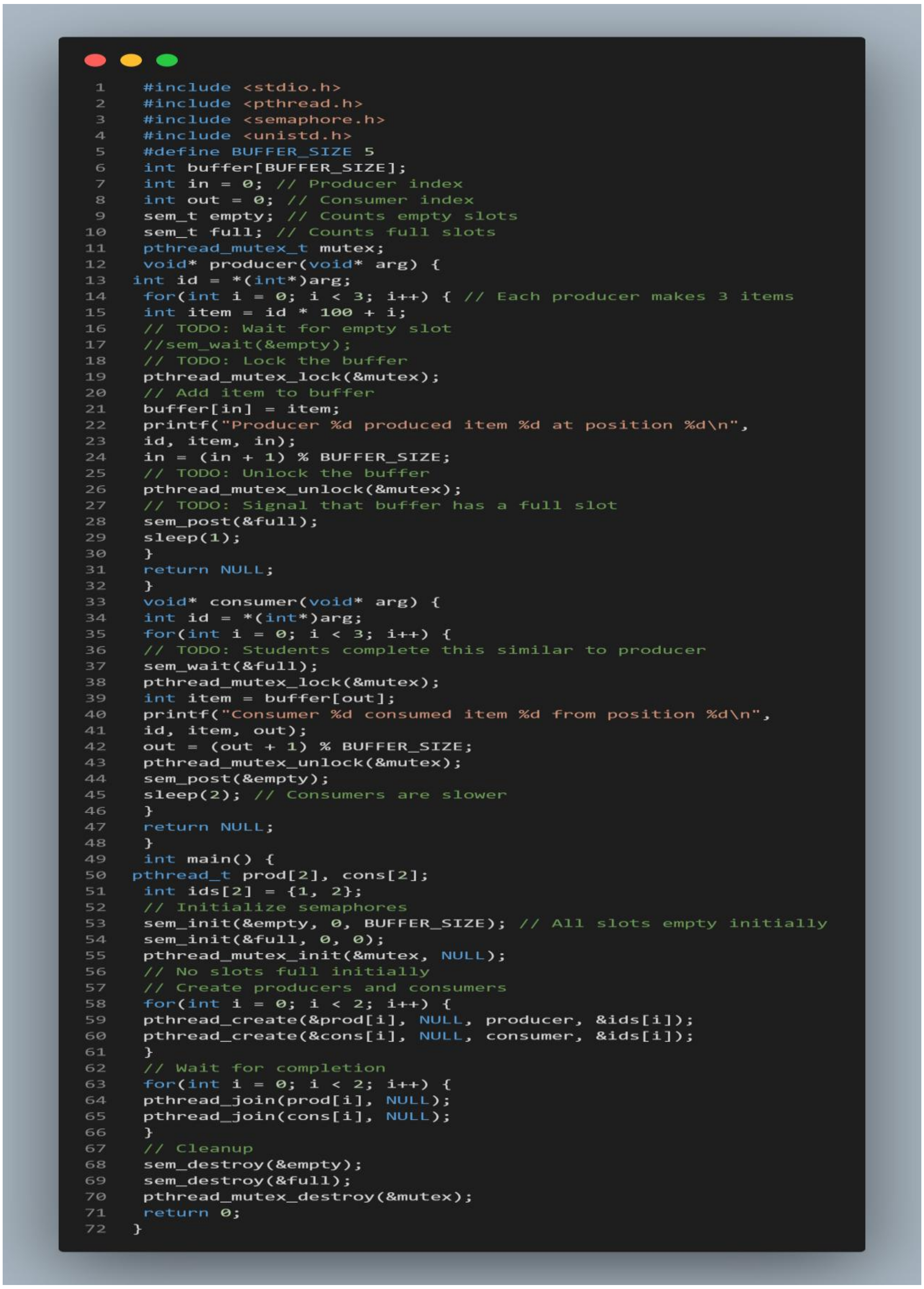
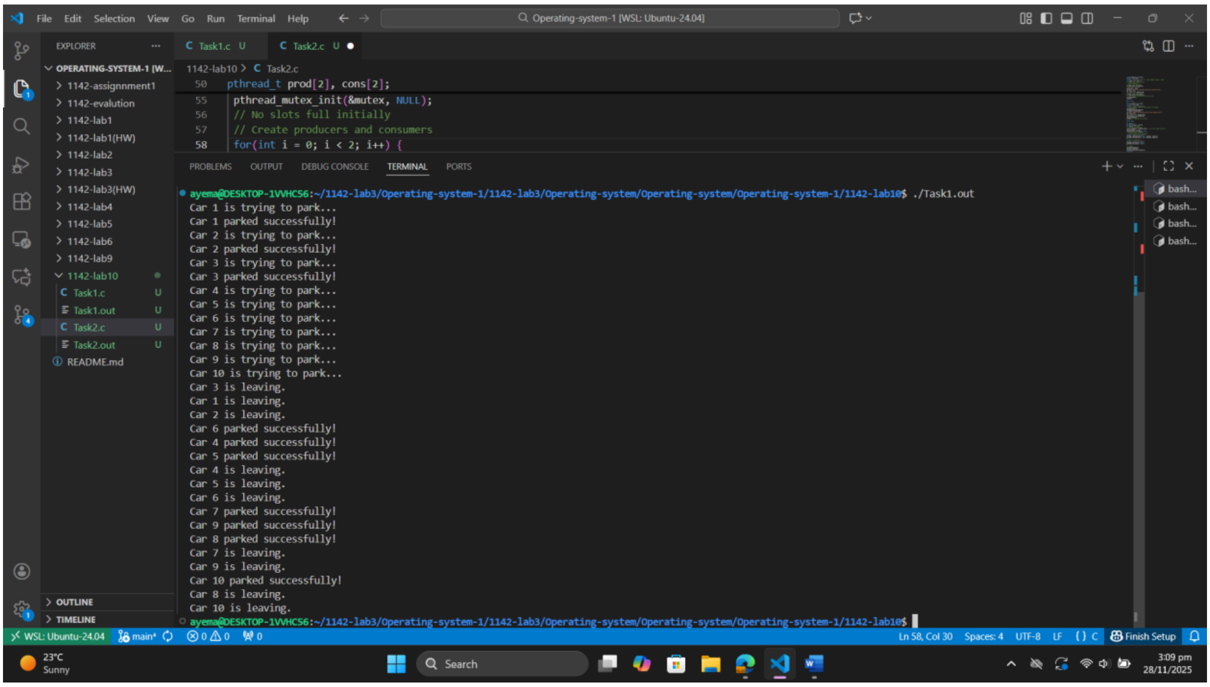
TASK 1:



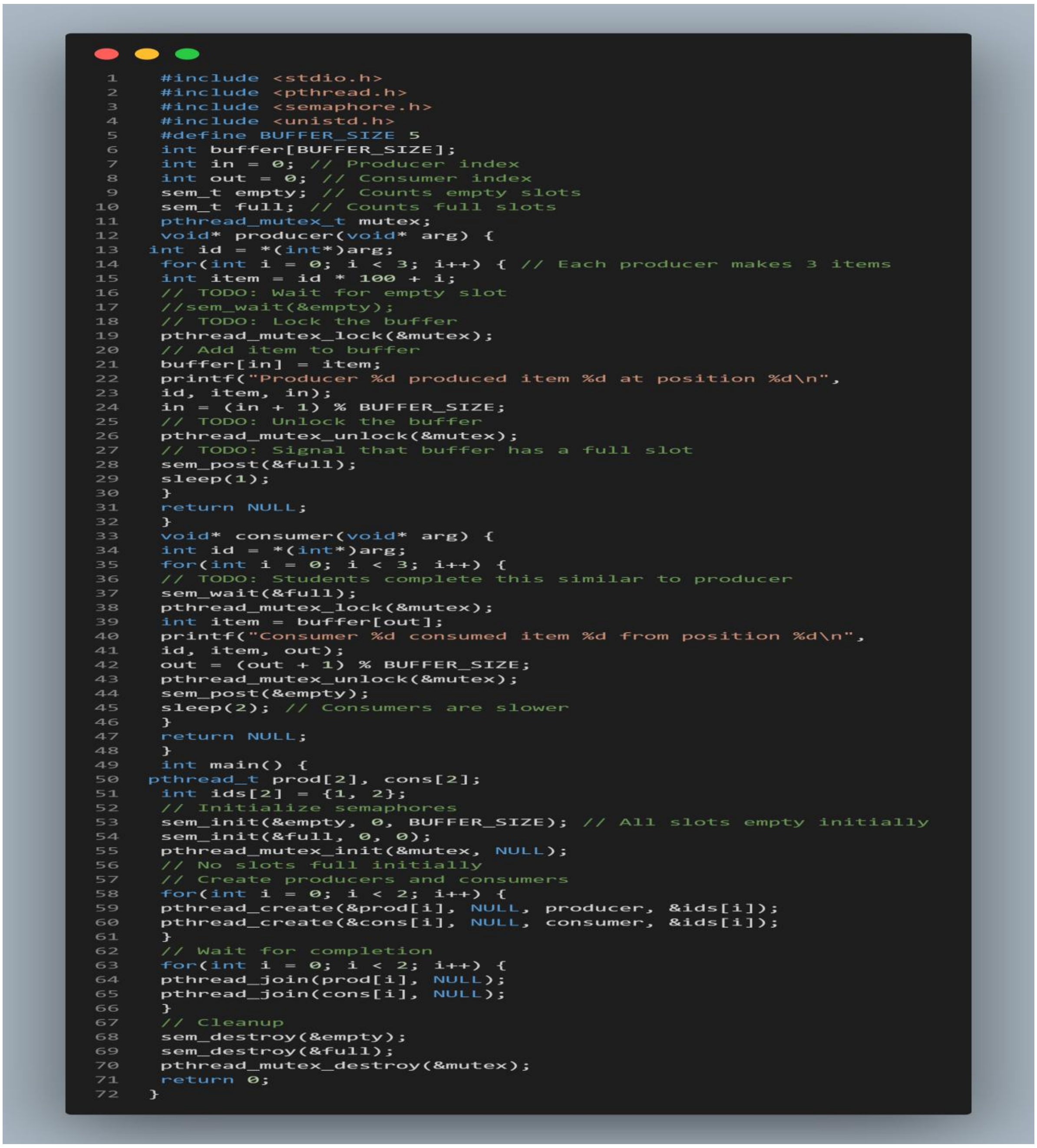
```
 #include <stdio.h>
 #include <pthread.h>
 #include <semaphore.h>
 #include <unistd.h>
 #define BUFFER_SIZE 5
 int buffer[BUFFER_SIZE];
 int in = 0; // Producer index
 int out = 0; // Consumer index
 sem_t empty; // Counts empty slots
 sem_t full; // Counts full slots
 pthread_mutex_t mutex;
 void* producer(void* arg) {
int id = *(int*)arg;
 for(int i = 0; i < 3; i++) { // Each producer makes 3 items
 int item = id * 100 + i;
 // TODO: Wait for empty slot
 //sem_wait(&empty);
 // TODO: Lock the buffer
 pthread_mutex_lock(&mutex);
 // Add item to buffer
 buffer[in] = item;
 printf("Producer %d produced item %d at position %d\n",
 id, item, in);
 in = (in + 1) % BUFFER_SIZE;
 // TODO: Unlock the buffer
 pthread_mutex_unlock(&mutex);
 // TODO: Signal that buffer has a full slot
 sem_post(&full);
 sleep(1);
 }
 return NULL;
 }
 void* consumer(void* arg) {
 int id = *(int*)arg;
 for(int i = 0; i < 3; i++) {
 // TODO: Students complete this similar to producer
 sem_wait(&full);
 pthread_mutex_lock(&mutex);
 int item = buffer[out];
 printf("Consumer %d consumed item %d from position %d\n",
 id, item, out);
 out = (out + 1) % BUFFER_SIZE;
 pthread_mutex_unlock(&mutex);
 sem_post(&empty);
 sleep(2); // Consumers are slower
 }
 return NULL;
 }
 int main() {
pthread_t prod[2], cons[2];
 int ids[2] = {1, 2};
 // Initialize semaphores
 sem_init(&empty, 0, BUFFER_SIZE); // All slots empty initially
 sem_init(&full, 0, 0);
 pthread_mutex_init(&mutex, NULL);
 // No slots full initially
 // Create producers and consumers
 for(int i = 0; i < 2; i++) {
 pthread_create(&prod[i], NULL, producer, &ids[i]);
 pthread_create(&cons[i], NULL, consumer, &ids[i]);
 }
 // Wait for completion
 for(int i = 0; i < 2; i++) {
 pthread_join(prod[i], NULL);
 pthread_join(cons[i], NULL);
 }
 // Cleanup
 sem_destroy(&empty);
 sem_destroy(&full);
 pthread_mutex_destroy(&mutex);
 return 0;
}
```



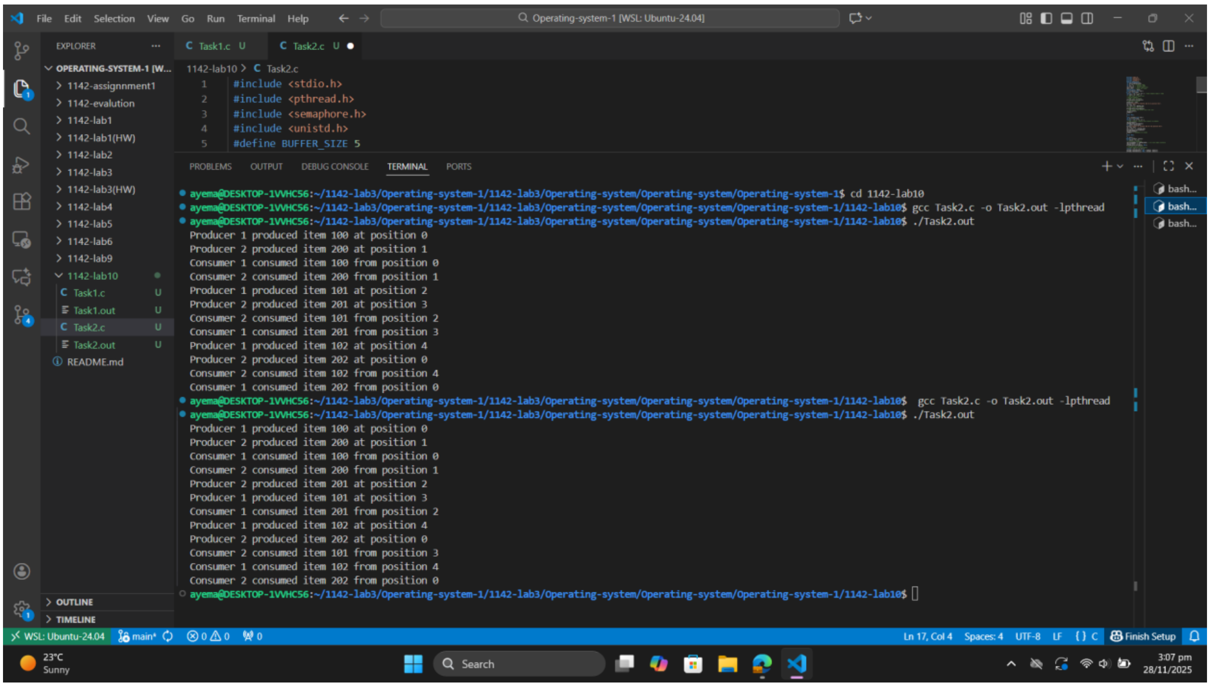
```c
pthread_t prod[2], cons[2];
    pthread_mutex_init(&mutex, NULL);
    // No slots full initially
    // Create producers and consumers
    for(int i = 0; i < 2; i++) {
```

```
ayema@DESKTOP-1VVHC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab10$ ./Task1.out
Car 1 is trying to park...
Car 1 parked successfully!
Car 2 is trying to park...
Car 2 parked successfully!
Car 3 is trying to park...
Car 3 parked successfully!
Car 4 is trying to park...
Car 5 is trying to park...
Car 6 is trying to park...
Car 7 is trying to park...
Car 8 is trying to park...
Car 9 is trying to park...
Car 10 is trying to park...
Car 3 is leaving.
Car 1 is leaving.
Car 2 is leaving.
Car 6 parked successfully!
Car 4 parked successfully!
Car 5 parked successfully!
Car 4 is leaving.
Car 5 is leaving.
Car 6 is leaving.
Car 7 parked successfully!
Car 9 parked successfully!
Car 8 parked successfully!
Car 7 is leaving.
Car 9 is leaving.
Car 10 parked successfully!
Car 8 is leaving.
Car 10 is leaving.
ayema@DESKTOP-1VVHC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab10$
```

Task 2:



```c
 #include <stdio.h>
 #include <pthread.h>
 #include <semaphore.h>
 #include <unistd.h>
 #define BUFFER_SIZE 5
 int buffer[BUFFER_SIZE];
 int in = 0; // Producer index
 int out = 0; // Consumer index
 sem_t empty; // Counts empty slots
 sem_t full; // Counts full slots
 pthread_mutex_t mutex;
 void* producer(void* arg) {
int id = *(int*)arg;
 for(int i = 0; i < 3; i++) { // Each producer makes 3 items
 int item = id * 100 + i;
 // TODO: Wait for empty slot
 //sem_wait(&empty);
 // TODO: Lock the buffer
 pthread_mutex_lock(&mutex);
 // Add item to buffer
 buffer[in] = item;
 printf("Producer %d produced item %d at position %d\n",
 id, item, in);
 in = (in + 1) % BUFFER_SIZE;
 // TODO: Unlock the buffer
 pthread_mutex_unlock(&mutex);
 // TODO: Signal that buffer has a full slot
 sem_post(&full);
 sleep(1);
 }
 return NULL;
 }
 void* consumer(void* arg) {
 int id = *(int*)arg;
 for(int i = 0; i < 3; i++) {
 // TODO: Students complete this similar to producer
 sem_wait(&full);
 pthread_mutex_lock(&mutex);
 int item = buffer[out];
 printf("Consumer %d consumed item %d from position %d\n",
 id, item, out);
 out = (out + 1) % BUFFER_SIZE;
 pthread_mutex_unlock(&mutex);
 sem_post(&empty);
 sleep(2); // Consumers are slower
 }
 return NULL;
 }
 int main() {
pthread_t prod[2], cons[2];
 int ids[2] = {1, 2};
 // Initialize semaphores
 sem_init(&empty, 0, BUFFER_SIZE); // All slots empty initially
 sem_init(&full, 0, 0);
 pthread_mutex_init(&mutex, NULL);
 // No slots full initially
 // Create producers and consumers
 for(int i = 0; i < 2; i++) {
 pthread_create(&prod[i], NULL, producer, &ids[i]);
 pthread_create(&cons[i], NULL, consumer, &ids[i]);
 }
 // Wait for completion
 for(int i = 0; i < 2; i++) {
 pthread_join(prod[i], NULL);
 pthread_join(cons[i], NULL);
 }
 // Cleanup
 sem_destroy(&empty);
 sem_destroy(&full);
 pthread_mutex_destroy(&mutex);
 return 0;
}
```



```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define BUFFER_SIZE 5
```

```
ayema@DESKTOP-1VVHC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1$ cd 1142-lab10
ayema@DESKTOP-1VVHC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab10$ gcc Task2.c -o Task2.out -lpthread
ayema@DESKTOP-1VVHC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab10$ ./Task2.out
Producer 1 produced item 100 at position 0
Producer 2 produced item 200 at position 1
Consumer 1 consumed item 100 from position 0
Consumer 2 consumed item 200 from position 1
Producer 1 produced item 101 at position 2
Producer 2 produced item 201 at position 3
Consumer 2 consumed item 101 from position 2
Consumer 1 consumed item 201 from position 3
Producer 1 produced item 102 at position 4
Producer 2 produced item 202 at position 0
Consumer 2 consumed item 102 from position 4
Consumer 1 consumed item 202 from position 0
ayema@DESKTOP-1VVHC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab10$  gcc Task2.c -o Task2.out -lpthread
ayema@DESKTOP-1VVHC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab10$ ./Task2.out
Producer 1 produced item 100 at position 0
Producer 2 produced item 200 at position 1
Consumer 1 consumed item 100 from position 0
Consumer 2 consumed item 200 from position 1
Producer 2 produced item 201 at position 2
Producer 1 produced item 101 at position 3
Consumer 1 consumed item 201 from position 2
Producer 1 produced item 102 at position 4
Producer 2 produced item 202 at position 0
Consumer 2 consumed item 101 from position 3
Consumer 1 consumed item 102 from position 4
Consumer 2 consumed item 202 from position 0
ayema@DESKTOP-1VVHC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab10$ 
```

Demonstrations:

As 3 items in code so total no is 6.

Total thread is 3

Mutex is 1.

Semaphore 2

If Item 4 items it produce total 8. Buffer size 5 Hain toh first 5 produce ho gai baki 3 blocked ho jai gai.Fir baad meh consumer consume karay ga aur fir Producer produce karkai sub consume or produce ho gai.