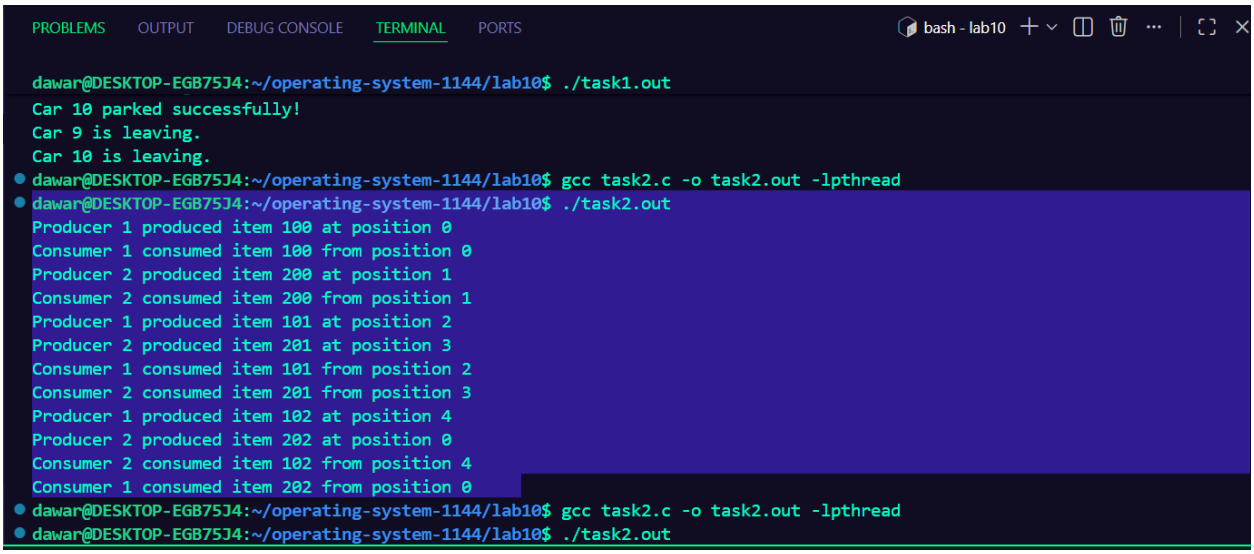
Lab10

Name: Dawar Abbas

Reg. No.: 1144

```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;
}
```



In this program we are using counting semaphore for producer consumer problem.

There are two producers and two consumers for a fixed 5 sized buffer. So when buffer is full we use the full thread to check and consume it by consumers when buffer is empty it will wait for it to produce first.

There are five steps both functions:

Producer:

- -1 from empty (wait(&empty))
- Lock
- Critical section
- Unlock
- +1 in full (post(&full))

Consumer:

- -1 from full (wait(&full))
- Lock
- Critical section
- Unlock
- +1 in empty (post(&empty))

If buffer is empty and nothing to consume then it will go to deadlock.