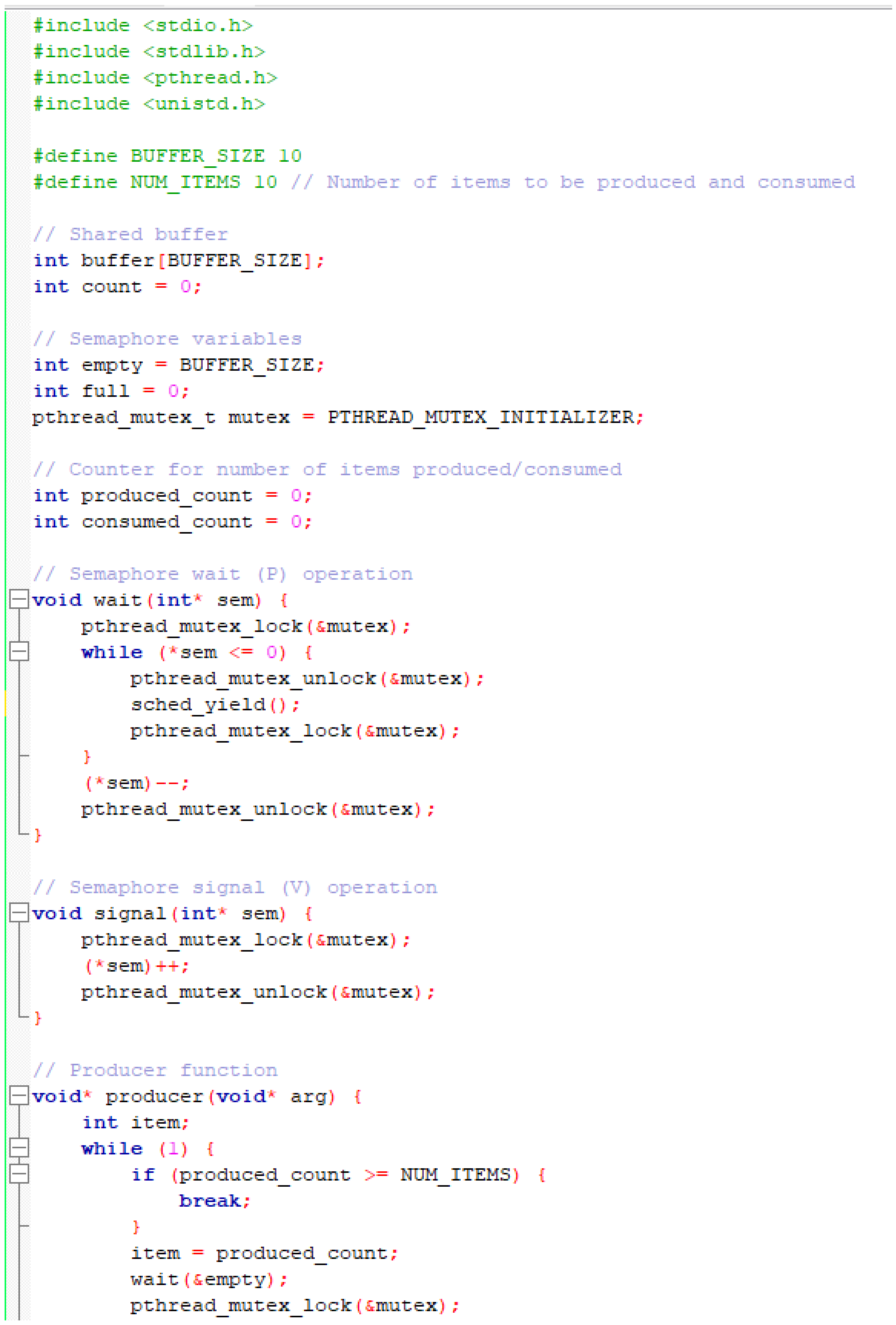
LAB PROGRAM 5

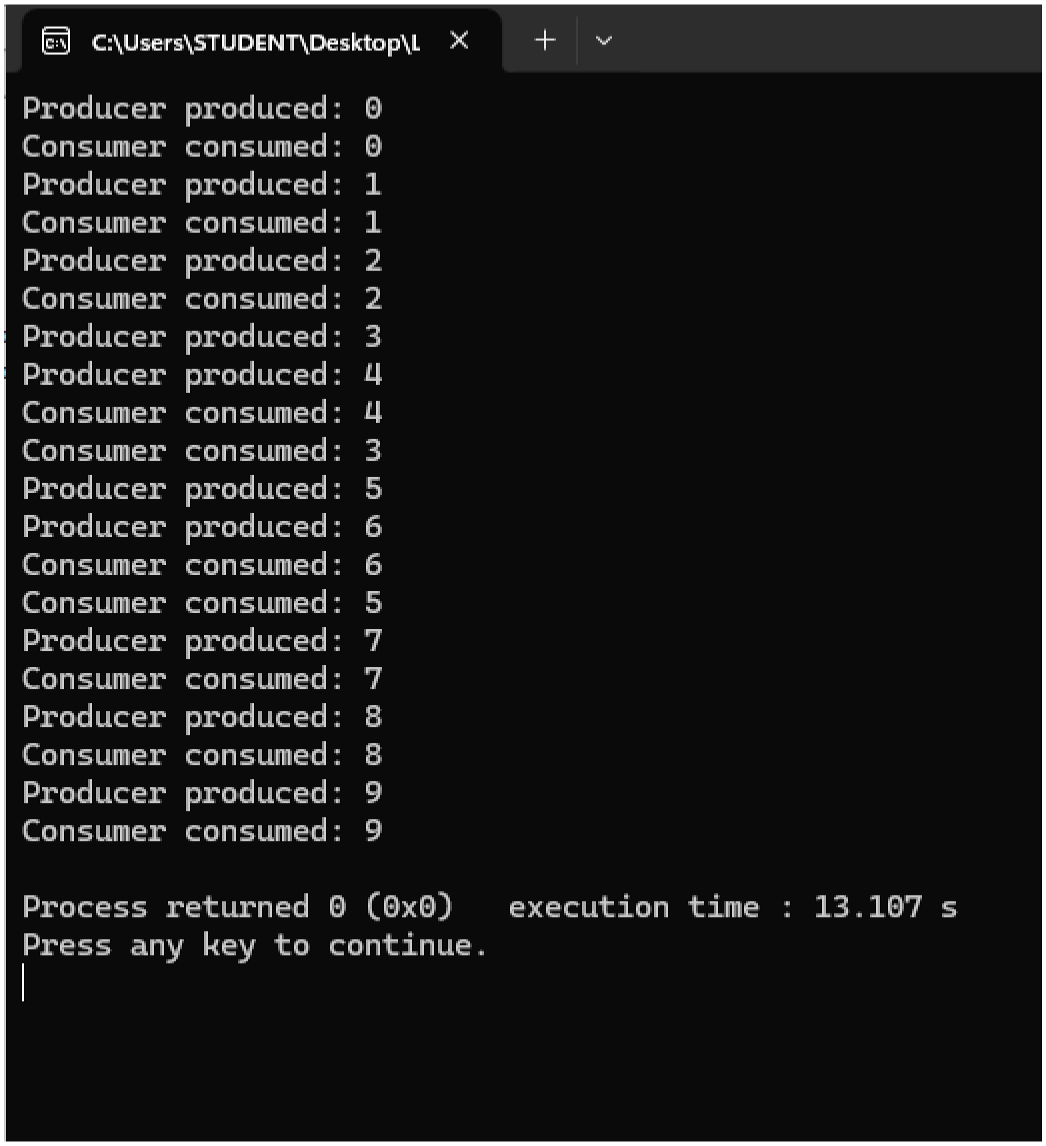
Write a C program to simulate producer-consumer problem using semaphores

INPUT



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

#define BUFFER_SIZE 10
#define NUM_ITEMS 10 // Number of items to be produced and consumed

// Shared buffer
int buffer[BUFFER_SIZE];
int count = 0;

// Semaphore variables
int empty = BUFFER_SIZE;
int full = 0;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

// Counter for number of items produced/consumed
int produced_count = 0;
int consumed_count = 0;

// Semaphore wait (P) operation
void wait(int* sem) {
    pthread_mutex_lock(&mutex);
    while (*sem <= 0) {
        pthread_mutex_unlock(&mutex);
        sched_yield();
        pthread_mutex_lock(&mutex);
    }
    (*sem)--;
    pthread_mutex_unlock(&mutex);
}

// Semaphore signal (V) operation
void signal(int* sem) {
    pthread_mutex_lock(&mutex);
    (*sem)++;
    pthread_mutex_unlock(&mutex);
}

// Producer function
void* producer(void* arg) {
    int item;
    while (1) {
        if (produced_count >= NUM_ITEMS) {
            break;
        }
        item = produced_count;
        wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[count] = item;
        count++;
        produced_count++;
        printf("Producer produced: %d\n", item);
        pthread_mutex_unlock(&mutex);
        signal(&full);
        sleep(rand() % 4);
    }
    return NULL;
}

// Consumer function
void* consumer(void* arg) {
    int item;
    while (1) {
        if (consumed_count >= NUM_ITEMS) {
            break;
        }
        wait(&full);
        pthread_mutex_lock(&mutex);
        count--;
        item = buffer[count];
        consumed_count++;
        printf("Consumer consumed: %d\n", item);
        pthread_mutex_unlock(&mutex);
        signal(&empty);
        sleep(rand() % 4);
    }
    return NULL;
}

int main() {
    pthread_t prod_thread, cons_thread;

    // Create the producer and consumer threads
    pthread_create(&prod_thread, NULL, producer, NULL);
    pthread_create(&cons_thread, NULL, consumer, NULL);

    // Wait for the threads to finish
    pthread_join(prod_thread, NULL);
    pthread_join(cons_thread, NULL);

    return 0;
}
```

OUTPUT


C:\Users\STUDENT\Desktop\L

```
Producer produced: 0
Consumer consumed: 0
Producer produced: 1
Consumer consumed: 1
Producer produced: 2
Consumer consumed: 2
Producer produced: 3
Producer produced: 4
Consumer consumed: 4
Consumer consumed: 3
Producer produced: 5
Producer produced: 6
Consumer consumed: 6
Consumer consumed: 5
Producer produced: 7
Consumer consumed: 7
Producer produced: 8
Consumer consumed: 8
Producer produced: 9
Consumer consumed: 9

Process returned 0 (0x0)   execution time : 13.107 s
Press any key to continue.
```