```c
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>


#define BUFFER_SIZE 5   // Size of buffer


int buffer[BUFFER_SIZE];

int in = 0, out = 0;


// Declare semaphores and mutex

sem_t empty;

sem_t full;

pthread_mutex_t mutex;


void *producer(void *arg)
{
    int item;
    for (int i = 1; i <= 10; i++) {
        item = i;  // produce an item (here just a number)

        sem_wait(&empty);          // Decrement empty count
        pthread_mutex_lock(&mutex);   // Lock critical section

        buffer[in] = item;
        printf("Producer produced item %d at buffer index %d\n", item, in);
        in = (in + 1) % BUFFER_SIZE;

        pthread_mutex_unlock(&mutex); // Unlock critical section
        sem_post(&full);           // Increment full count
```

```c
        sleep(1); // simulate production time
    }
    return NULL;
}


void *consumer(void *arg)
{
    int item;
    for (int i = 1; i <= 10; i++) {
        sem_wait(&full);          // Decrement full count
        pthread_mutex_lock(&mutex);   // Lock critical section

        item = buffer[out];
        printf("Consumer consumed item %d from buffer index %d\n", item, out);
        out = (out + 1) % BUFFER_SIZE;

        pthread_mutex_unlock(&mutex); // Unlock critical section
        sem_post(&empty);         // Increment empty count

        sleep(2); // simulate consumption time
    }
    return NULL;
}


int main()
{
    pthread_t producerThread, consumerThread;

    // Initialize semaphores and mutex
    sem_init(&empty, 0, BUFFER_SIZE);
```

```c
    sem_init(&full, 0, 0);

    pthread_mutex_init(&mutex, NULL);


    // Create threads

    pthread_create(&producerThread, NULL, producer, NULL);

    pthread_create(&consumerThread, NULL, consumer, NULL);


    // Wait for threads to finish

    pthread_join(producerThread, NULL);

    pthread_join(consumerThread, NULL);


    // Destroy semaphores and mutex

    sem_destroy(&empty);

    sem_destroy(&full);

    pthread_mutex_destroy(&mutex);


    printf("\nAll items produced and consumed successfully.\n");

    return 0;
}
```

output

Producer produced item 1 at buffer index 0

Consumer consumed item 1 from buffer index 0

Producer produced item 2 at buffer index 1

Producer produced item 3 at buffer index 2

Consumer consumed item 2 from buffer index 1

Producer produced item 4 at buffer index 3

Consumer consumed item 3 from buffer index 2

Producer produced item 5 at buffer index 4

Consumer consumed item 4 from buffer index 3

Producer produced item 6 at buffer index 0

Consumer consumed item 5 from buffer index 4

Producer produced item 7 at buffer index 1

Consumer consumed item 6 from buffer index 0

Producer produced item 8 at buffer index 2

Consumer consumed item 7 from buffer index 1

Producer produced item 9 at buffer index 3

Consumer consumed item 8 from buffer index 2

Producer produced item 10 at buffer index 4

Consumer consumed item 9 from buffer index 3

Consumer consumed item 10 from buffer index 4

All items produced and consumed successfully.