```
#include <stdio.h>
#include <pthread.h>
#include <stdbool.h>
#include <semaphore.h>
#include "queue.h"
#define randdelay 6

int buffer_size = 4;
sem_t full, empty, mutex;
int buf = 0;

void producerFunction(int id)
{
        printf("PRODUCER %d : Started\n", id);
        while(true)
        {
                //producing item
                int x = (rand() % 10) +1;
                printf("PRODUCER %d : Produced Item - %d\n", id, x);

                sem_wait(&empty);
                sem_wait(&mutex);

                //critical section
                buf++;
                insert(x);
                printf("PRODUCER %d : Placed Item, Buffer Size - %d\n", id, buf);

                sem_post(&full);
                sem_post(&mutex);
                sleep(rand() % randdelay);
        }
}
void consumerFunction(int id)
{
        printf("CONSUMER %d : Started\n", id);
        while(true)
        {
                sem_wait(&full);
                sem_wait(&mutex);

                //critical section
                buf--;
                int x = removeitem();
                printf("CONSUMER %d : Removed Item, Buffer Size - %d\n", id, buf);

                sem_post(&empty);
                sem_post(&mutex);

                //consume item
                printf("CONSUMER %d : Consumed Item - %d\n", id, x);
                sleep(rand() % randdelay);
        }
}
int main()
{
        int numThreads = 5;
        pthread_t producer[numThreads], consumer[numThreads];

        int res1 = sem_init(&mutex, 0, 1);
        int res2 = sem_init(&full, 0, 0);
        int res3 = sem_init(&empty, 0, buffer_size);

        if(res1 != 0 || res2 != 0 || res3 != 0)
        {
                printf("Semaphore Initialization Failed..\n");
                return 1;
        }

        ///creating threads
        for(int i=0;i<numThreads;i++)
        {
                res1 = pthread_create(&producer[i], NULL, (void *) producerFunction, i+1);
                res2 = pthread_create(&consumer[i], NULL, (void *) consumerFunction, i+1);

                if(res1 != 0 || res2 != 0)
                {
                        printf("Thread Creation Failed..");
                        return 1;
                }
        }
```

```
        //joining threads back
        for(int i=0;i<numThreads;i++)
        {
                res1 = pthread_join(producer[i], NULL);
                res2 = pthread_join(consumer[i], NULL);

                if(res1 != 0 || res2 != 0)
                {
                        printf("Thread Join Failed..\n");
                        return 1;
                }
        }

        //destroying semaphores
        res1 = sem_destroy(&mutex);
        res2 = sem_destroy(&full);
        res3 = sem_destroy(&empty);

        if(res1 != 0 || res2 != 0 || res3 != 0)
        {
                printf("Semaphore Destroy Failed..\n");
                return 1;
        }

        return 0;
}

/* OUTPUT -
PRODUCER 1 : Started
PRODUCER 1 : Produced Item - 4
PRODUCER 1 : Placed Item, Buffer Size - 1
PRODUCER 3 : Started
PRODUCER 3 : Produced Item - 8
PRODUCER 3 : Placed Item, Buffer Size - 2
CONSUMER 1 : Started
CONSUMER 1 : Removed Item, Buffer Size - 1
CONSUMER 1 : Consumed Item - 4
CONSUMER 2 : Started
CONSUMER 2 : Removed Item, Buffer Size - 0
PRODUCER 4 : Started
PRODUCER 5 : Started
PRODUCER 5 : Produced Item - 7
CONSUMER 2 : Consumed Item - 8
CONSUMER 3 : Started
PRODUCER 4 : Produced Item - 6
CONSUMER 4 : Started
PRODUCER 5 : Placed Item, Buffer Size - 1
PRODUCER 4 : Placed Item, Buffer Size - 2
CONSUMER 2 : Removed Item, Buffer Size - 1
CONSUMER 2 : Consumed Item - 7
CONSUMER 5 : Started
PRODUCER 2 : Started
PRODUCER 2 : Produced Item - 8
CONSUMER 3 : Removed Item, Buffer Size - 0
CONSUMER 3 : Consumed Item - 6
PRODUCER 2 : Placed Item, Buffer Size - 1
CONSUMER 4 : Removed Item, Buffer Size - 0
CONSUMER 4 : Consumed Item - 8
PRODUCER 3 : Produced Item - 7
PRODUCER 3 : Placed Item, Buffer Size - 1
CONSUMER 5 : Removed Item, Buffer Size - 0
CONSUMER 5 : Consumed Item - 7
PRODUCER 3 : Produced Item - 3
PRODUCER 3 : Placed Item, Buffer Size - 1
CONSUMER 5 : Removed Item, Buffer Size - 0
CONSUMER 5 : Consumed Item - 3
PRODUCER 2 : Produced Item - 8
PRODUCER 2 : Placed Item, Buffer Size - 1
PRODUCER 4 : Produced Item - 7
PRODUCER 4 : Placed Item, Buffer Size - 2
CONSUMER 2 : Removed Item, Buffer Size - 1
CONSUMER 2 : Consumed Item - 8
CONSUMER 3 : Removed Item, Buffer Size - 0
CONSUMER 3 : Consumed Item - 7
PRODUCER 5 : Produced Item - 4
PRODUCER 5 : Placed Item, Buffer Size - 1
PRODUCER 4 : Produced Item - 6
CONSUMER 5 : Removed Item, Buffer Size - 0
CONSUMER 5 : Consumed Item - 4
PRODUCER 4 : Placed Item, Buffer Size - 1
PRODUCER 4 : Produced Item - 3
```

```
PRODUCER 4 : Placed Item, Buffer Size - 2
PRODUCER 4 : Produced Item - 10
PRODUCER 4 : Placed Item, Buffer Size - 3
PRODUCER 1 : Produced Item - 4
PRODUCER 1 : Placed Item, Buffer Size - 4
PRODUCER 5 : Produced Item - 2
CONSUMER 2 : Removed Item, Buffer Size - 3
PRODUCER 2 : Produced Item - 3
CONSUMER 2 : Consumed Item - 6
CONSUMER 3 : Removed Item, Buffer Size - 2
CONSUMER 3 : Consumed Item - 3
PRODUCER 2 : Placed Item, Buffer Size - 3
PRODUCER 5 : Placed Item, Buffer Size - 4
CONSUMER 1 : Removed Item, Buffer Size - 3
CONSUMER 1 : Consumed Item - 10
CONSUMER 4 : Removed Item, Buffer Size - 2
CONSUMER 4 : Consumed Item - 4
*/
```