Assignment4 mutex and semaphore

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.concurrent.Semaphore;

class ProducerConsumer {
   private static final int BUFFER_SIZE = 5;
   private static Queue<Integer> buffer = new LinkedList<>();
   private static Semaphore mutex = new Semaphore(1);
   private static Semaphore empty = new Semaphore(BUFFER_SIZE);
   private static Semaphore full = new Semaphore(0);
   public static void main(String[] args) {
      Thread producer = new Thread(new Producer());
      Thread consumer = new Thread(new Consumer());
      producer.start();
      consumer.start();
   }
static class Producer implements Runnable {
      public void run() {
         try {
            for (int i = 0; i < 10; i++) {
               empty.acquire(); // wait for empty slot
               mutex.acquire(); // enter critical section
              buffer.add(i);
               System.out.println("Produced: " + i);
              mutex.release(); // exit critical section
               full.release(); // signal that buffer is not empty
            }
         } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
         }
```

```
    }
  }
  static class Consumer implements Runnable {
    public void run() {
      try {
        for (int i = 0; i < 10; i++) {
          full.acquire(); // wait for full slot
          mutex.acquire(); // enter critical section
          int item = buffer.remove();
          System.out.println("Consumed: " + item);
          mutex.release(); // exit critical section
          empty.release(); // signal that buffer has empty slots
        }
      } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
      }
    }
  }
}
```

OUTPUT: