```
1   #James Roesemann
2   #CSCI 375
3   #Operating Systems
4   #Project 1
5   #due 6/19/2018
6   #written in python
7
8   """
9   This program is a implmentation of the producer consumer problem.
10  the user will be asked to input an integer to determine the buffer size and the
    counter limit.
11  the program will then call two threads, producer amd consumer
12  producer will add random integers to the buffer, so long as the buffer is not
    full. when added it will decrement the produceLimit
13  consumer will remove interger from the buffer, in order, so long as the buffer is
    not empty. when removed it will decrement consumeLimit
14  when produceLimit =0 the producer thread will end
15  when consumeLimt=0 the consumer limit will end
16
17  i'm still learning python and haven't figured out how to pass specfic variables
    to a thread. I relize it's a bit sloppy but i'm going to use a few global
    variables in the threads to demonstrate the producer/consumer problem.
18  """
19
20  import time
21  import random
22  import threading
23
24
25  #producer is the producer thread. recives no input, initlized with .start()
26  class producer( threading.Thread):
27          def run(self):
28                  global bufferList
29                  global produceLimit
30                  #while produceLimit is > 0, aquire the lock from the condition
    object bufferLock
31                  while produceLimit>0:
32                          bufferLock.acquire()
33                          #if bufferList=bufferSize, then the buffer is full. wait
    untill notified by the consumer
34                          if len(bufferList)==bufferSize:
35                                  bufferLock.wait()
36                          #generate a random integer between 0 and 1000, append it
    to the end of the buffer.
37                          nextProduced=random.randint(0,1000)
38                          bufferList.append(nextProduced)
39                          print 'Produced:', nextProduced
40                          #increment produceLimit, notify the consumer that
    producer has produced to the buffer, release bufferLock
41                          produceLimit-=1
42                          bufferLock.notify()
43                          bufferLock.release()
44
45  #consumer is the consumer thread. recives no input. initlized with .start()
46  class consumer(threading.Thread):
47          def run(self):
48                  global bufferList
49                  global consumeLimit
50                  #while consumerLimit is > than zero, acuire the lock from the
    condition object bufferLock
51                  while consumeLimit>0:
52                          bufferLock.acquire()
```

```
53                          #if bufferList is empty,  wait untill notified by producer
54                          if not bufferList:
55                                  bufferLock.wait()
56                          #now that bufferList has something in it, consume the
    first element of the list. decrement consumeLimit by 1
57                          nextConsumed = bufferList.pop(0)
58                          print 'Consumed:', nextConsumed
59                          #deincrement consumeLimit, notify producer that consumer
    as consumed from bufferlist and release bufferLock
60                          consumeLimit-=1
61                          bufferLock.notify()
62                          bufferLock.release()
63
64  #begining of program
65  random.seed()
66  bufferLock=threading.Condition()
67  bufferSize=input('What would you like the buffer size to be? enter a number: ')
68  produceLimit=input('What would you like the counter Limit to be? enter a number:
    ')
69  consumeLimit=produceLimit
70  bufferList=[]
71
72  producer().start()
73  consumer().start()
```