# SOURCE CODE

from criticalSection import CriticalSection

from boundedBuffer import BoundedBuffer

import threading

def test\_critical\_section():

cs = CriticalSection()

threads = []

for i in range(10):

t = threading.Thread(target=cs.critical\_section, args=(i,))

threads.append(t)

t.start()

for t in threads:

t.join()

def test\_bounded\_buffer():

bb = BoundedBuffer(1)

producer\_threads = threading.Thread(target=lambda: [bb.producer(i) for i in range(10)])

consumer\_threads = threading.Thread(target=lambda: [bb.consumer() for \_ in range(10)])

producer\_threads.start()

consumer\_threads.start()

producer\_threads.join()

consumer\_threads.join()

if \_\_name\_\_ == '\_\_main\_\_':

print('Testing critical section')

test\_critical\_section()

print('Testing bounded buffer')

test\_bounded\_buffer()

# Output Screenshot

Text

Description automatically generated

# Thoughts

The participation activity went smoothly. I don’t know if I had any particular insights, but seeing the code running helped me see how semaphores work. One thing I changed from the way we did things in class was using a deque() in bounded buffer to add items to the left and remove them from the right, so they are first-in, first-out.