CS 305 :: Operating Systems

Homework Three (50 points)

Question: In this homework, you will implement the bounded-buffer problem for a producer-consumer scenario utilizing semaphores in python/ java/ another programming language. Both producer and consumer work on shared data, where producer adds items in the buffer, and consumer removes items for use. However, they will follow the following three conditions:

1. They cannot work simultaneously on the data as they both have to perform a write operation.

2. Consumer cannot consume data item if the buffer is empty.

3. Producer cannot add a new data item if the buffer is full.

You will need three semaphores to synchronize the write and wait operations on the shared buffer. One is to make sure that producer is waiting on a full buffer, one for consumer waiting on an empty buffer, and the last is to make sure that only the producer or consumer is working on the shared buffer data.

Now, in the main function

1. Create three semaphores and initialize them with appropriate values.

2. From the main function, create a thread for the producer, another thread for the consumer, and manage them.

3. Either put the shared buffer, all the semaphores in the global scope, or pass them to the threads. Each Item will be an integer number in the shared buffer.

4. In the main function, use a variable (n) to manipulate the number of maximum items in the shared buffer. Set the value of n to 20. The producer will add 100 items, and the consumer will consume 100 items in total, so they will use and reuse the buffer 5 times.

Algorithm Design: I want you to think critically on how you can design the algorithm for the producer and consumer thread on what should be sequences of these semaphores, what should be the initialization values for each of the semaphores, when a thread should call the wait, and when to call the signal, etc. Design your algorithm, and then implement them. Now you can see whether your algorithm works perfectly by looking at input-output sequences, whether the producer is waiting on the full buffer, the consumer waiting on the empty buffer, the output sequence, etc. If there is some problem, think again, and update your algorithm. Do not look for the solution on the internet or the book. If you are stuck somewhere, consult with me after class/ office hours.

What to print:

1. Print out each time the producer or consumer will call wait and signal function. One print statement before the call and one after the function returns. For example:

```
print("Producer is waiting on write semaphore")

wait(write)

print("Producer got access on write semaphore")
```

2. Each time Producer adds an item or consumer removes an item, print out the item, and index value. For example: “Produced added item with value 5 at index 2”, or “Consumer removed item with value 4 from index 0”.

Implementation hints for python:

1. Use the “threading” module to introduce threading and semaphore in your program.

2. Use sp = threading.Semaphore(p) to create a new semaphore with initial value p.

3. Use sp.acquire() and sp.release() to wait and signal on semaphore.

Implementation hints for java:

1. Import the “java.util.concurrent.Semaphore” module.

2. Use Semaphore sp = new Semaphore(p) to create a new semaphore with initial value p.

3. Use sp.acquire() and sp.release() to wait and signal on semaphore.

What to Submit: Submit a zip file contacting the following two items:

1. A python/ java file containing code for the question.

2. A doc/pdf file containing the following:

    a. Screenshots of your program output.

    b. A short description of your algorithm design for the producer and consumer thread.