LAB 1

Harsh Shah

List of potential problems and faulty situations that may arise without using any coordination or synchronization mechanisms:

1. **Buffer overflow**: The producer may try to produce more items even if the buffer becomes full resulting in a buffer overflow problem. Thus, the producer must be asked to wait until the consumer consumes an item and a slot becomes available.

2. **Empty buffer**: The consumer may try to consume from an empty buffer which results in a problem and hence must not be allowed to access the buffer if it is empty.

3. **Deadlock**: One of the ways to prevent the producer from accessing a full buffer, and the consumer from accessing an empty buffer, is by using sleep () and notify () methods. But implementing this may lead to a race condition which in turn leads to deadlock, when one of the notify () calls gets lost and both sleep forever.

4. **Dirty writes**: Consider that there are more than one producers and both decide to produce at the same time. They check the buffer and find a slot empty. Now both the producers produce in the same slot which is undesirable. Similarly, two consumers may try to consume from the same slot.

Output:

harshyshah@ufl.edu

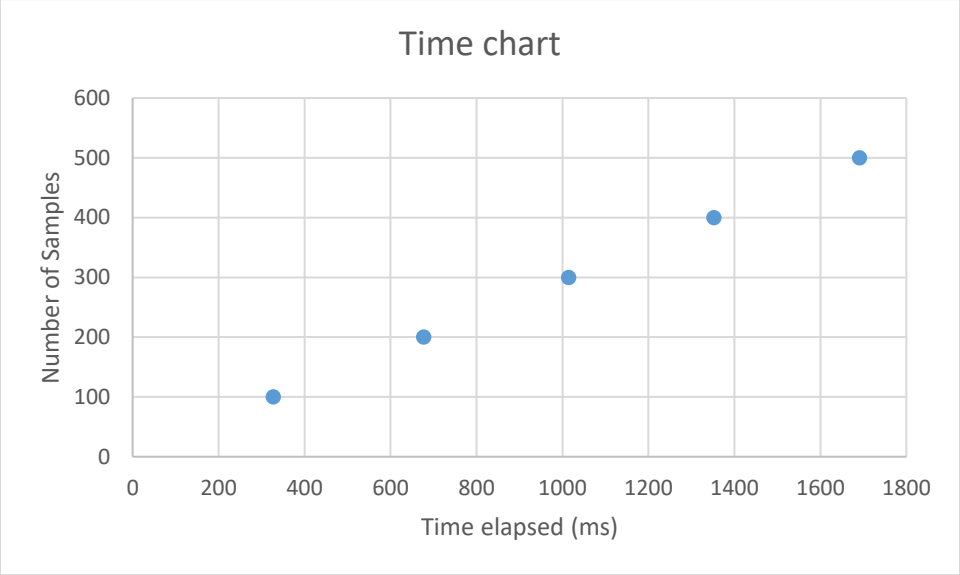
LAB 1

Harsh Shah

Timing graphs:

C} Mutex

Number of Samples	Time Elapsed (ms)
100	331
200	676
300	1014
400	1353
500	1690



D} Counting Semaphores

Number of Samples	Time Elapsed (ms)
100	327
200	677
300	1014
400	1352
500	1691

harshyshah@ufl.edu

LAB 1

Harsh Shah

Conclusion:

As seen from the above tables, there is not much difference in the timing values between mutex code and counting semaphore code and as a result, the values are so close that they almost intersect on the graph. The reason being that there are only two processes: a producer and a consumer. The difference would have probably been significant if there were more than one producers and/or more than one consumers.

Also, we noticed that the consumer consumed faster than the producer produced after a certain number of samples. If we would have used the print (I/O) statement outside the critical section (which is what we should do, as the consumer should be allowed to consume when the producer goes for I/O operations), we would notice that the consumer prints that it has consumed the data even before the producer printed its output entirely. This shows that the consumer works faster even when both the processes have the same priority.

So, it would be a good practice to use only one mutex if there are only two processes i.e. one producer and one consumer. Additionally, counting semaphores must be used if there are more than two processes.

harshyshah@ufl.edu