Theory Assignment-1

(PCP)

Name- Souvik Sarkar

Roll- CS23MTECH02001

3.7)

This is an illustration of a situation that illustrates this issue:

- Thread A uses the enq() method to enqueue an item with a value of x.(it only execute till line no 9 and then preempted)
- Thread B tries to use the deq() method to dequeue an item right away.
- Thread B attempts to fetch the item after noticing that the head is pointing to the same index that Thread A was just queuing.(this thread execute the whole code)

The issue, though, is that the enqueued item is not visible. There is no guarantee that the new value x has been completely written to memory and made visible to Thread B, even in cases where Thread A has correctly enqueued the item. This implies that rather than reading the actual enqueued value x, Thread B might read an old value or null.

The linearizability criteria is broken by this inconsistent visibility of operations. Before Thread B's dequeue operation can start, Linearizability would need to ensure that Thread A's enqueue operation is finished and visible to all threads.

3.10)

i) Give an execution showing that the linearization point for () cannot occur at line 15.

Let A and B, two parallel enq() operations, both with tail equal to 0 at the beginning:After executing tail.getAndIncrement(), operation A yields tail = 1. After executing tail.getAndIncrement(), operation B yields tail = 2.Assume that the tail increment of operation B finishes before the tail increment of operation A:When Operation B wants to queue an item, it performs items[1].set(y).At this stage, operation B has stored the itsitem and reserved slot 1 in the array.Items[0].set(x) is executed with its item in Operation A.Operation B's item was saved before operation A's, despite the fact that operation A's tail increment happened before operation B's. This execution shows that because the order of item storage does not match the order of tail increments, the linearization point for enq() cannot be at line 15 (the increment of tail).

ii) Give an execution showing that the linearization point for () cannot occur at line 15.

Let A, B, and C be two concurrent enq operations and one concurrent deq operation, with A and B beginning with tail equal to 0:Operation A handles the tail.obtainAndIncrement(), which yielded tail = 1. Operation B executes the tail.obtainAndIncrement(), which yielded tail

= 2.When Operation B wants to queue an item, it performs items[1].set(y).At this stage, operation B has stored the itsitem and reserved slot 1 in the array.Items[0].set(x) is executed with its item in Operation A.Subsequently, C initiates deq, deq x, then y.Despite the fact that operation B set the value y before operation A did, we receive x before y in the deqprocess. This execution shows that because the order of item storage does not match the order of tail increment, the linearization point for enq() cannot be at line 16.

iii) Since these are the only two memory accesses in enq(), we must conclude that () has no single linearization point. Does this mean () is not linearizable?

It is feasible to determine whether or not the enq() and deq methods of the Herlihy-Wing queue are linearizable based on the description and sample code that have been provided.The enq() techniqueThe two main tasks of the enq() method are to store an item in the array and increment the tail index. The state of the queue is not entirely consistent during the window of time between the increment and the item storage since these two actions are not atomic.The enq() procedure cannot be linearized because the enq() operation does not linearize at a single fixed location in the code. An operation must appear to take effect instantly at a specified point in time between its invocation and response in order to be linearizable. In this instance, the absence of a distinct linearization point denotes a failure to meet the linearizability condition.The deq approachThere isn't a single fixed linearization point in the deq approach either. After reading the tail value, iteratively exchanging null with the current contents of the array as it moves through it. Once more, there is no precise moment at which the deq process is claimed to linearize.Like the enq() method, the deq method may not be linearizable if there isn't a distinct linearization point.In conclusion, the Herlihy-Wing queue implementation's enq() and deq procedures cannot be linearized based on the information supplied. The absence of clearly defined linearization points for operations in these methods suggests a breach of the criteria for linearizability.