CS1760: Multiprocessor Synchronization Due 27 November 2023

Homework 6

Professor: Maurice Herlihy *TA: cs1760tas@lists.brown.edu*

Problem 1. Consider the unbounded lock-based queue's deq() method in Figure 1. Is it necessary to hold the lock when checking that the queue is not empty? Explain.

```
1  public T deq() throws EmptyException {
2      T result;
3      deqLock.lock();
4      try {
5        if (head.next == null) {
6          throw new EmptyException();
7        }
8        result = head.next.value;
9        head = head.next;
10     } finally {
11       deqLock.unlock();
12     }
13     return result;
14   }
```

Figure 1: The UnboundedQueue<T> class: the deq() method.

Problem 2. Consider the linearization points of the enq() and deq() methods of the lock-free queue:

1. Can we choose the point at which the returned value is read from a node as the linearization point of a successful deq()? Explain.
2. Can we choose the linearization point of the enq() method to be the point at which the tail field is updated, possibly by other threads? Explain.

Problem 3. Modify the unbounded lock-free stack from the texbook and lectures to work in the absence of a garbage collector. Create a thread-local pool of preallocated nodes and recycle them. To avoid the ABA problem, consider using the AtomicStampedReference<T> class from java.util.concurrent.atomic, which encapsulates both a reference and an integer *stamp*.