ECE/CS 5510 Multiprocessor Programming Final Exam

Mincheol Sung
ID: 9060-28300

December 9, 2018

Problem 1.

```
———— Thread 0 ————                           ———— Thread 1 ————
1: flag[0] = true;                           1: flag[1] = true;
2: label[0] = max(label[0],label[1])+1;      2: label[1] = max(label[0],label[1])+1;
3: While(flag[1] &&                          3: While(flag[0]
4: (label[0], 0) > (label[1], 1));           4: && (label[1], 1) > (label[0], 0));
```

If line 1 is executed after line 4 and/or updated label is not visible to other thread at line 2 due to delayed write, both of them can enter their critical section.

Problem 2.

As a method A is linearizable if and only if all its nested are linearizable, there is no need to modify the linearization definition.

Problem 3.

Yes. Prof Bumstead is correct. the linear point should be at line 28 when the deq() gets the sentinel as its first. Then, the test of first == head.get() is passed at line 32 such that the deq eventually throws the exception.

Problem 4.

To recycle memory, objects should be tracked to avoid being garbage collected. We can maintain a free list containing unused queue nodes for each thread. Instead of allocating a new node, pop a node from the free queue. Push a node into the free list instead of freeing the node. In addition, need to use CAS operations and check the case where the free list is empty. To avoid ABA problem, we can leverage AtomicStampedReference to tag a time stamp to each pointer.

Problem 5.

The professor is wrong. Between bucket(myBucket) == NULL and curr.key == key, add() can insert sentinel node. So getSentinel() needs to check whether curr.key is equal to the sentinel key of myBucket.

Problem 6.

Add() will work since its linearization point is when the bottom level is linked.
Remove() will work since its linearization point is when the bottom level next reference is marked.