Linked Lists: The Role of Locking

Question 1. Implement the set data type using linked lists with optimistic synchronization (i.e. search without acquiring any locks at all. If the method finds the sought-after component, it locks that component, and then checks that the component has not changed in the interval between when it was inspected and when it was locked)

Question 2. In the optimistic algorithm:

a) Give a scenario where a thread is forever attempting to delete a node.

b) Is it okay if the `add()` method locks only *pred*? If so, why?

c) Suppose the `contains()` method locks no entries, returns true if it observes the value, and false otherwise. Either explain why this alternative is linearizable, or give a counterexample showing it is not.

Note: While determining linearization points, you need to find them for both successful (returns true) and unsuccessful (returns false) operations.

Question 3. Implement the set data type using linked lists with lazy synchronization (i.e. the task of removing a component from a data structure can be split into two phases: the component is logically removed simply by setting a tag bit, and later, the component can be physically removed by unlinking it from the rest of the data structure)

Question 4. In the lazy algorithm:

a) Would the algorithm work, i.e., remain linearizable, if we marked a node as removed simply by setting its next field to null? Why or why not?

b) Can the validation method be simplified by dropping the check that pred.next is equal to curr (while preserving linearizability)? Why or why not?

c) Does locking only the *pred* or only the *curr* work in remove()? Justify your answer regarding whether these approaches are linearizable. If not, provide a counterexample.

d) What if, in remove(), I drop the physical deletion and only perform logical deletion? Is this still linearizable? Justify your answer.

Note: For all the above questions, try to reproduce counterexamples (if they exist) using your implementation of lazy synchronization. This may require adding "sleeps" in the code of the set methods.