Question 1

* 1. Search: (assuming we search for node i)  
     1. lock node 1
     2. lock node 2
     3. unlock node 1
     4. lock node 3
     5. unlock node 2  
        … only node i-1 is locked
     6. lock node i (required node is found)
     7. unlock node i-1
  2. insert: (assuming we insert after node i)
  3. remove:

1. Observation 1: Basically, the observation means that one thread (we’ll denote as B) cannot overtake another thread (we’ll denote as A). This is true because when a thread reads a node, it locks it, meaning no other thread can read the same node. Since a thread (A) always hold the lock for at least one node at any given time, in order for another thread (B) to overtake it (A), it (B) needs to go through a node that is currently locked by the first thread (A). This is obviously not possible since a locked node cannot be read and thus cannot be passed.  
     
   Observation 3: When a thread is using a prime candidate *p*, all prime numbers less then *p* have already been used by the thread. Together with observation 1 we can infer that all non-prime numbers, which are a multiplication of the used primes, were deleted (by either this thread or another). is not divisible by any previous number (or else it would have been deleted)   
   Observation 2: Observation 3 is prime is only divisible by can only be removed by
2. the work needed to be done by the algorithm is deleting all non-prime numbers.   
   if all threads run the exact same (single-threaded with hand-over-hand synchronization) code, the locks becomes the bottlenecks. Only one thread (we’ll denote as T) will succeed in locking the first node (we’ll denote as N) before all else, All other threads will have to wait until the node unlocks. Hand-over-hand dictates that T will always hold the lock to at least one node at a time, so other threads can’t overtake T because they need to read N before continuing. T will always be first to arrive to a non-prime number, thus T will be the only thread to delete any number. NOT COMPLETE