Programming Assignment 2

Due date: Thursday, March 23rd by 2:00 PM

Design and Implementation of Concurrent Linked List:
Design a coarse-grained locking linked list, as described in Chapter 9 of the textbook. Save this version of the linked list and call it BLOCKING_LIST. Transform your algorithm into a lock-free algorithm and call it NONBLOCKING_LIST. Your implementation should be based on Tim Harris' paper "A Pragmatic Implementation of Non-Blocking Linked Lists." You may also use Section 9.8 of the textbook for reference.

Evaluate your approach by executing performance comparison of your BLOCKING_LIST and NONBLOCKING_LIST. In your benchmark tests, vary the number of threads from 1 to 32 and produce graphs where you map the total execution time on the y-axis and the number of threads on the x-axis. Produce at least 3 different graphs representing different ratios of the invocation of insert, delete, and find. For example, your different ratios could be:

- Mixed: 33% insert, 33% delete, 34% find
- Write-dominated: 50% insert, 50% delete, 0% find
- Read-dominated: 15% insert, 5% delete, 80% find

To avoid "polluting" your results with the overhead of memory management (the standard malloc and free don't scale well), you should have each thread pre-allocate its own supply of nodes, which it can keep on a private list when they are not in the list. Finally, to avoid pollution from calls to the pseudo-random (the standard rand isn't even thread-safe), you should pregenerate enough random integers to drive the choice between insert, delete, and find for the entire test run.

Provide a brief summary of your approach and an informal statement reasoning about the correctness and efficiency of your design.

Use either C or C++ for this assignment and provide a ReadMe file with instructions explaining how to compile and run your program.