CSE 411: Project 2

Including Memory Reclamation in Synchrobench's Lazy Linked-List

Due date: Friday October 29 (Groups of max 2)

- Download Synchrobench from the following link, and follow the instructions to setup and run it: https://github.com/gramoli/synchrobench

- Part 1:
  - Check the lazy-list C code and make sure you understand it well.
  - Make a code review and report your notes on:
    - Readability of code and documentation.
    - Potential needs for code refactoring (either in the lazy list code or in Synchrobench in general).
    - Potential thread safety issues.

- Part 2:
  - This part is to assess performance and correctness consequences of reclaiming memory *unsafely.*
    - Modify lazy-list so that a remove operation reclaims the node's memory (using *delete* – no object pooling) right after physically deleting it.
    - Compare the performance of this *unsafe* version with the original implementation (without reclamation) in different workloads. Show your results in the report.
    - Build an artificial scenario (using delays and deterministic operations for example) that clarifies how this unsafe implementation can raise run-time errors. Explain the scenario and its results in the report.
- Part 3:
  - It is now the time to build a correct reclamation approach. You should implement and compare the two approaches discussed in lectures (show your results in the report):
    - C++ smart pointers
    - Epoch-Based Reclamation
  - Extra credits (up to 15% of the total grade) will be given to any optimizations on top of the base-line approaches discussed in lectures, as well as any other investigated (and implemented/compared) reclamation technique.