Ari ‘aith’ I.

| | Coarse-grained | Reader-Writer | Swaptop (R-W) |
|---|---|---|---|
| Total calls | 398,574 | 951,728 | 795,162 |
| Variance (relative) | 0.744 | 0.224 | 0.262 |

In part 3 I implemented 3 thread-safe, conflict-free stacks using c++ locks. The first version uses a coarse-grained lock, which entails a single lock for all 3 operations [push, pop, peek]. This implementation is inefficient because, as we saw in part 2, peek() is thread-safe when there are any number of readers and 0 writers. For this reason, the Reader-Writer lock performed almost 3x as many calls as the coarse-grained lock. Unlike part 2, I did not implement a fair Reader-Writer lock. This starved the writers tremendously, and helps explain why the total call count is so high.

Implementing Swaptop turned out to be trickier than at first glance. It adds a swaptop() function that changes the data held by the final node in the linked-list. I leveraged the efficiency of the shared-lock by first checking if the data that was called to be swapped was actually different from the data already in the list. If so, then the function would then shift to an exclusive lock. The tricky part was handling the edge case where the final node in the list would be deleted in between the change in locks. For this reason I needed to verify the final node twice within swaptop().