Lab 1 Report

Caden Keese

Task 2

```
Thread 1:|>------ add 1 ------<| |>----- rmv 1 ------<|
Thread 2:---------|>------ add 1 ------<| | ------ add 2 -------|
```
you can have something like this where thread 1 does not finish adding 1 before thread 2 starts adding 1 you can end up with duplicate data, so at the end you will have one extra "1" in your set than there should be

```
Thread 1:|>------ add 1 ------<| |>----- rmv 1 ------<|
Thread 2:---------|>------ rmv 1 ------<| | ------ add 2 -------|
```
you can also have something like this where thread 1 starts removing 1 before thread 2 has finished and depending on implementation you can end up with a null pointer exception as the value has been delete by thread 2 while thread 1 is accessing it or a different error.

Task 3

The linearization occurs when the operation is called as the global lock blocks all other operations from occurring after that point.

Task 4

the linearization occurs when the data structure is actually modified after the traversal.

For task 4 I was able to get fine grained locking to work for add and contain but not remove. with remove I had trouble with it not correctly linearizing with an add right before, (ie. t1:add(20),t1:add(30),t2:rmv(10)) after around 10 items were in the list and only for items in the later parts of the list, I was able to get around this problem by using a shared lock block remove operations during add operations or contain operations.
I was trying to do something like below, if you have any suggestions in why this didn't work that would be appreciated

```
given: head->[10]->[20] -> [36]->[49]->[53]->[55]
with a lock for the head pointer and a lock for every item
add 50:
        lock head pointer lock
        lock 10 lock
        lock 20 lock
        unlock head pointer lock
        lock 36 lock
        unlock 10 lock
        lock 49 lock
        unlock 20 lock
        lock 53 lock
        unlock 36 lock
        [49]->[50]
        [50]->[53]
        unlock 53
        unlock 49
        => head->[10]->[20]->[36]->[49]->[50]->[53]->[55]
remove 50:
        lock head pointer lock   <> head🔒->[10] ->[20] ->[36] ->[49] ->[50] ->[53] ->[55]
        lock 10 lock             <> head🔒->[10]🔒->[20] ->[36] ->[49] ->[50] ->[53] ->[55]
        lock 20 lock             <> head🔒->[10]🔒->[20]🔒->[36] ->[49] ->[50] ->[53] ->[55]
        unlock head pointer lock <> head ->[10]🔒->[20]🔒->[36] ->[49] ->[50] ->[53] ->[55]
        lock 36 lock             <> head ->[10]🔒->[20]🔒->[36]🔒->[49] ->[50] ->[53] ->[55]
        unlock 10 lock           <> head ->[10] ->[20]🔒->[36]🔒->[49] ->[50] ->[53] ->[55]
        lock 49 lock             <> head ->[10] ->[20]🔒->[36]🔒->[49]🔒->[50] ->[53] ->[55]
        unlock 20 lock           <> head ->[10] ->[20] ->[36]🔒->[49]🔒->[50] ->[53] ->[55]
        lock 50 lock             <> head ->[10] ->[20] ->[36]🔒->[49]🔒->[50]🔒->[53] ->[55]
        unlock 36 lock           <> head ->[10] ->[20] ->[36] ->[49]🔒->[50]🔒->[53] ->[55]
        [49]->[53]               <> head ->[10] ->[20] ->[36] ->[49]🔒       ->[53] ->[55]
        unlock 49                <> head ->[10] ->[20] ->[36] ->[49]        ->[53] ->[55]
        => head->[10]->[20]->[36]->[49]->[53]->[55]
```