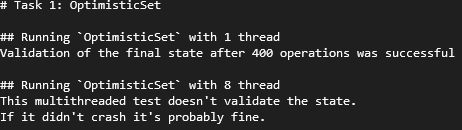
**Concurrent Data Structures Lab Assignment 2**

Bernat Xandri Zaragoza

1. **Task 1:** Optimistic synchronization.

We implemented the Optimistic synchronization in the list-based set. It was important to keep in mind corner cases like adding or deleting the first element of the list, because of the dealing with an empty list. It required to add some extra parts to the code.



1. **Task 2:** Lazy synchronization.

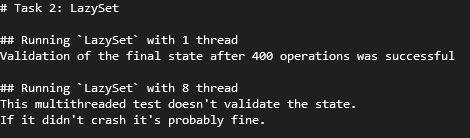
Up to this point I have always worked with the set as a non-ordered set. For this lazy implementation though, and since we’re using the locate function, it would make more sense to switch into an ordered set. This is also how it described in the slides.

I’m unsure if we were expected to always work with an ordered set, but as far as I could read, this was not specified anywhere in the lab instructions, and given that we didn’t get any grading or feedback yet, I consider it correct and therefore did the previous exercise as unordered.

For this implementation, we had to switch some of the evaluations to make sure the set is ordered now.

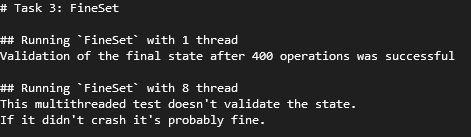
It also took quite some work to manage to write a locate function that was general enough for all other methods, and at the same time took into account all corner cases.

I had to change the template so that it returns 2 pointers.



1. **Task 3:** Fine-Grained Set.

I decided to use the code provided with the correction of Lab1, since mine was incorrect.



1. **Task 4:** Experiment.

For this task I had to update the makefile, since the flags were not working correctly. I had to include to use C++ version 17.

Also, after setting up everything to be able to run this task, I encountered that despite passing all the tests from previous tasks, the algorithms from previous tasks wouldn’t run in this benchmarking. I was getting core dumped errors, dreadlocks, and just early finalizations. I tried running the task using the debugger gdb, but when running in the debugger I wouldn’t encounter any of these errors. After reading in the internet a bit about this, I found out that this is a known event in concurrent programming, where the debugger handles the memory in a more cautious way than the usual compiler, and therefore some data races just don’t appear in the debugger.

After trying many things, I couldn’t find a way to fix it, and it only runs correctly in the debuggers.

1. **Task 5:** Fine-Grained Multiset.

I implemented the same code from the fine-grained set, only changing the method add so that it can always add the element.

