We have successfully implemented a splay tree sequentially and set up our benchmarking environment where we can evaluate the correctness and speedup of our code. During our source reading, we came across an implementation that works top-down, where splaying rotations start at the root and propagate down until hitting the target node. We finished a trivial concurrent implementation where we lock the entire tree when a thread is accessing it. However, when it comes to fine-grained locking, we have success on various test cases but are coming across a race condition bug that causes some values to be deleted (despite the trace having no operations to delete values).

As of right now we are on pace with the schedule that we provided in the proposal. We are however having a bug in our fine-grained locking implementation. Once we have this bug fixed, we think that we should be able to produce all of the deliverables that we mentioned in the proposal. Based on how long these two implementations have taken and how long the testing and data gathering has taken, we think that we will most likely not be able to do any of the extra "nice to haves". In the unlikely case that we do have extra time, we think that we will most likely spend the time on more data collection, testing, and analysis. Based on our progress, this is our list of goals that we hope to hit for the poster session:

1. Finished implementation for sequential, fine-grained locking, and lock free splay tree
2. Finish debugging bugs that will certainly come
3. Collect all data needed for presentation and analysis
4. Produce graphs that help illustrate the speedups
5. Create final presentation with explanation of code and any other relevant information

At the poster session, we intend on showing the performance of our concurrent splay tree implementation. We can utilize data traces, showing the time spent to complete a given task and graphs of speedup on different test cases. To better show the algorithm behind our implementation, we will also show excerpts of code (along with explanations) to show the differences between the three different implementations. Finally, we can demo our implementation on a very simple trace with two processors, printing the tree structure after concurrent inserts or accesses.

Our most prevalent concern is debugging and testing our code. Because there is no checker or tester given to us, we have to make our own. Thus, it becomes a big concern on whether our tests are robust enough to determine that our code is completely correct. We spent a good amount of our time just making tests for the sequential and fine-grained locking implementations in order to convince ourselves of their correctness. We are even more concerned about testing the correctness of the lock free structure and we believe that this may be a very time-consuming process.