Kyle Zalewski

CS202

Efficiency Writeup

My biggest challenge in this assignment was finding the most sensible ways to use operator overloading with my classes. There were several times in the process where I felt it would be best to implement certain overloads as member functions but that was not possible. For instance, in my App class, for reading in information from the file, I thought it would be best to use dynamic binding to choose the best version of the insertion operator to use at run time. This would allow different types of data to be read in based on the object type the pointer was pointing to. This turned out to be impossible, since the insertion operator needs to be implemented as a friend (which cannot be virtual).

The extraction operator was an additional challenge. I had difficulty getting the call in main to actually use my overload, which resulted in an output file that was just filled with memory addresses for objects in the tree. The cause of this will be a continuing investigation. I believe it has everything to do with my overload signature, and the compiler simply isn’t recognizing that overload as a viable option for the call in main.

Other operators went much more smoothly. The comparison operators were quite straightforward, and referencing class notes helped me to bring implementations of those operators down from three lines to one each (which helped immensely with readability, considering there were three versions of each operator overload, and six operators total). I found that many of the operators I was writing functions for ended up being already implemented in other functions that I used, so those overloads ended up being fancy looking wrappers that gave me an additional and more elegant way to call functions from within my other implementations (outside of the given class).

As for the data structure, I ended up going with a regular binary search tree to focus on implementing the overloads rather than the data structure implementation. This allowed me to pay more attention to the skill the program was really designed to build. The most challenging portion of this implementation was removal of a node that has two children. The inorder successor must replace the App in question, and without a look back, it was tricky to reconnect the children of the inorder successor to its parent. Eventually, after breaking down the problem into much smaller sub-problems and individual cases, I was able to get it figured out.