Kyle Zalewski

CS202

Project 1 Postmortem

This project helped me to better understand inheritance. It made me critically think about what exactly I had access to when dealing with a derived class object, and helped me to understand why a particular class should be derived from another. I believe my final design was fairly straightforward and uncomplicated, though at times I felt I maybe could have employed a bit more complexity to achieve a final product that was more feature-rich.

As for object-orientation, I believe I achieved my desired result. The vast majority of my main cpp file consisted of creating objects and manipulating those, as well as a little bit of user input/output. I believe in this regard I achieved a good amount of modularity, and created code that is both readable and adaptable. I did have to hard-code some portions of the program due to time constraints, which if I was attempting to make the most universal classes possible, would not be acceptable.

In a similar future project, I would probably attempt to make a circular linked list for the user’s entire route, rather than use linear linked lists for the two halves of the trip. I think this would both exercise some programming skills and be a bit more succinct for actual real-world use, considering most people are visiting the mountain, rather than just moving there in a one-way trip. I think that lists in some capacity are the right choice for a program of this nature, but there are a few different implementations that might make the solution more elegant, or at least more realistic.

I ran into a few issues with memory leaks when I thought my program was complete. I think in the future I would avoid these pitfalls by paying more attention to memory management from the beginning. At the point I discovered these issues I was already so far into coding the hierarchy that chasing down every allocation and every delete was a much larger task than it would have been had I started earlier. I mentioned in my debugger writeup that my incremental approach to writing the program saved me from endless debugger use, but unfortunately the same cannot be said about memory leak testing. In future projects I will keep in mind the need to continuously rigorously test my code all along the development process to ensure that memory leaks are not a giant mountain to tackle at the end of the programming process.