Project 1
Binary Search Tree Project
CS 241

Julio Berina
jmberina@cpp.edu
1/30/18

The project involves reading a sequence of integers from a prompt, creating a binary search tree out of those values, and interacting with that tree through a loop with menu options. Before list the options, the pre-order, in-order, and post-order traversal of the tree is printed on the screen. Options include inserting and deleting values as well as finding the predecessor and successor of a node. Also, H and E are other non-tree interacting commands that list all possible options and exit the program respectively. The prompt continues until the user prompts to exit by entering "E".

The way the loop is set up is that the command takes a string and splits the input into a String array. The second argument of the array is parsed to an integer and applied to the corresponding tree method from the command (input[0]). All tree interactions involve the use of recursion. When inserting and deleting, the tree is printed via an in-order traversal, or a message is displayed stating that the value already exists or doesn't exist. When finding the predecessor or successor of a node, the result is simply displayed, or a message is displayed saying that there is none.

For testing the project, a good place to start for me was simply using the sample integer sequence provided in proj1.pdf. Anytime I implemented one of the methods for the binary search tree, I compiled and ran the program right away with the sample data. I made sure I inserted and/or deleted as much as I could, which led me to spotting more faults in my implementation. I did the same for finding the predecessor and successor of a node. I made sure that it was as dummy proof as possible by testing out the sample data as well as providing strange combinations of inputs to make odd-looking trees.

What I got the most out of doing this project is that testing code is extremely important when it comes to developing programs. The program may show that everything is completely fine, but it behaves strangely once it's given a bizarre set of inputs. I had this problem for the most part with my implementation of deleting a node. When testing my first implementation, I tried to delete a value, and it ended up deleting a total of four nodes which called for a remake. My second implementation was shorter and better, and I was able to come up with pretty much a decent dummy-proof method after fixing some minor errors.