Nicholas Skinner

CS260

Binary Tree/Binary Search Tree

Design:

While creating this assignment, I looked over numerous guides on the internet, and sketched out at least 4 different approaches on how to implement this. I followed some online guides fairly closely, and think I got a desirable result. The base tree was easy enough, parent references two children, unless there are not enough children to go around. The really challenging part is the formula behind finding where to place the children, using the parent to compare what should insert into what location. I couldn't wrap my head around this design for a while until I asked how my friend who had taken this class before me had approached it, after his walthrough/explanation it made a lot more sense using the parents as a reference.

Testing:

Testing for this post-completion had been fairly limited, I only have one input field, that I dont use input validation for, and just have to ensure that my positions in my Tree are acceptable. I went on to add a few additional test cases to check behavior:

Test case 1 - Correct input

Worked appropriatly, printed what was needed

Test case 2 - incorrect input

ran into an exception, could not complete and crashed.

Test case 3 - duplicate values

Inserted them to appropriate location, and then printed

Test case 4 - no values

It worked, could still search, but no values turn up

Reflection:

This was a very challenging assignment, my end goal was clear, and it was (for the most part) clear how to make the tree. The difficult part was really implementing the logic to make it a binary search tree, I had to consult many online guides to figure out what I was doing, and where to go next with my design. I think this assignment was a great practice on how to use my external resources and apply them for maximized results. while the others needed some understanding of how it works to implement, this one felt a lot more complicated (even though it ended up taking up fewer lines) and had me doing a lot more research into the logic of how the function should be working.