Project #6
CS 2210 – Fall 2021
Sam DeCook

I. Requirements: Restate the problem specification, and any detailed requirements
Make a ListBinaryTree class and 3 classes for each type of tree traversal. For the traversals, just display the value stored in the node. The node will be a sibling tree node (STNode), meaning it has pointers to its parent, left child, and sibling. The right node should have a null sibling pointer.

II. Design: How did you attack the problem? What choices did you make in your design, and why? Show class diagrams for more complex designs.
I implemented each method of the ListBinaryTree (which implemented the BinaryTree interface). For the more complicated methods, I first thought through each possibility with the help of a visual on my whiteboard and then wrote the code to take care of each case.

III. Security Analysis: State the potential security vulnerabilities of your design. How could these vulnerabilities be exploited by an adversary? What would be the impact if the vulnerability is exploited?
I don't know of any security vulnerabilities this design has. It doesn't access anything which could give deeper access and it prints out its output, so it has a limited scope and reach.

IV. Implementation: Outline any interesting implementation details.
At first, I implemented isInternal by returning !isExternal. However, there are a lot of if-statements needed to directly implement isExternal. You need to make sure both children are null, so first you test if the left child is null, and if it isn't, you have to test if the sibling (right child) is null. By reversing it (implementing isExternal by !isInternal), I was able to write much less and much more readable code. I just need to check if the left child was null. If it wasn't, the node was internal, and vice versa.

V. Testing: Explain how you tested your program, enumerating the tests if possible. Explain why your test set was sufficient to believe that the software is working properly, i.e., what were the range of possibilities of errors that you were testing for.
I tested each method and each type of return it could give (eg. true and false for Boolean return types). Then I tested the errors it threw. I'm pretty confident my test set it sufficient because the function to build the tree was given to us, and thus works.

VI. Summary/Conclusion: Present your results. Did it work properly? Are there any limitations? If it is an analysis-type project, this section may be significantly longer than for a simple implementation-type project.
Everything works properly. I don't think there are any limitation to the code I wrote, mainly because it doesn't do a lot to begin with.

VII. Lessons Learned: List any lessons learned. What might you have done differently if you were going to attack this again.
Take better notes on in-class guidance for the project. I thought my notes were enough, but there were a few situations where they weren’t. Other than that (and what I talked about Implementations) I wouldn’t change what I did. This project wasn’t very complicated code-wise.

My code compiled properly and ran as expected and produced correct output.