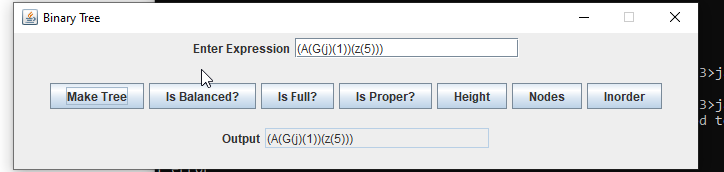
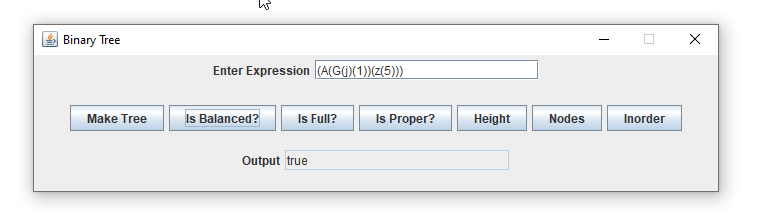
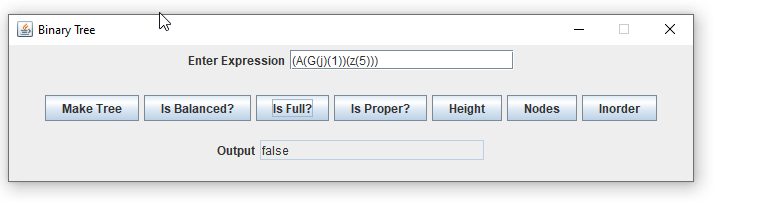
Test Case 1



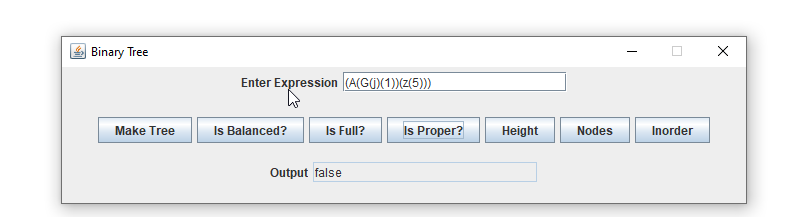
Is Balanced



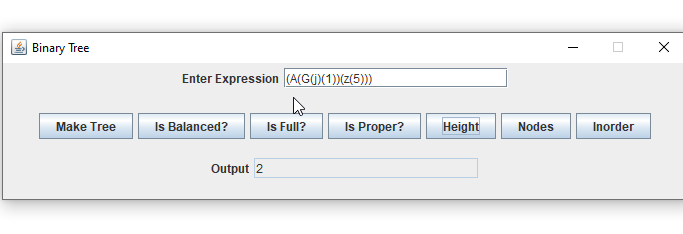
Is Full



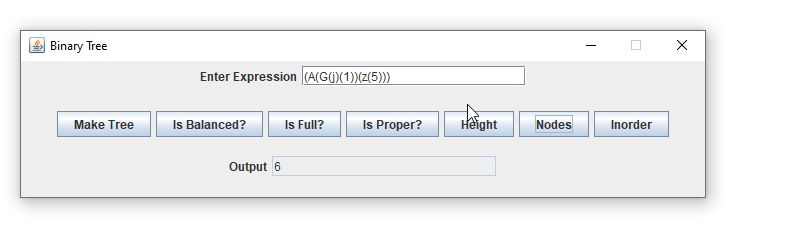
Is Proper



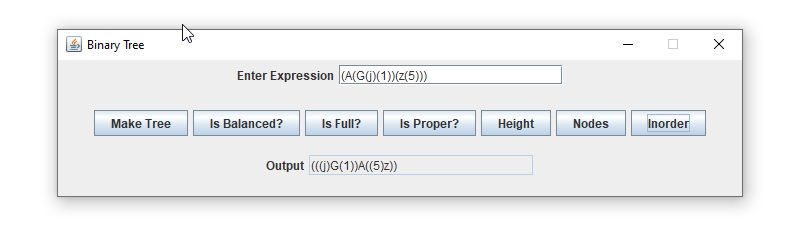
Height



Nodes

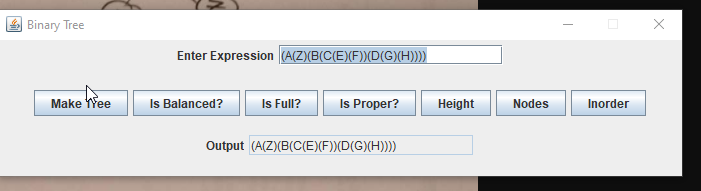


In Order

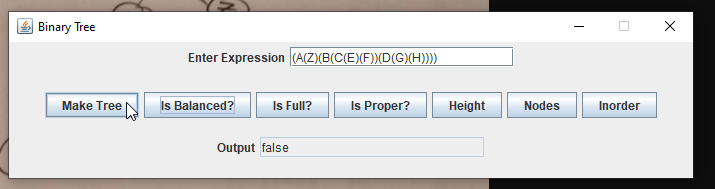


Test Case 2: (A(Z)(B(C(E)(F))(D(G)(H))))

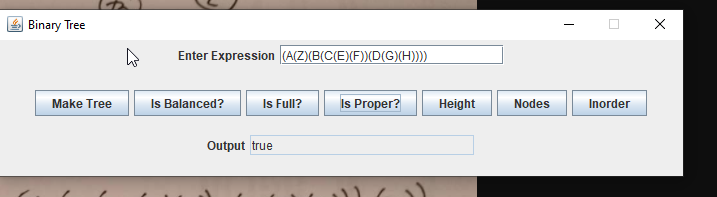
Make Tree



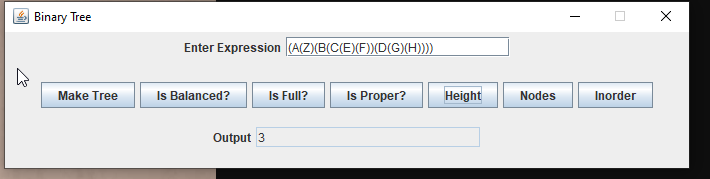
IS Balanced



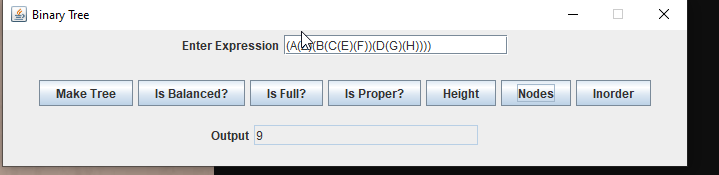
Is Proper



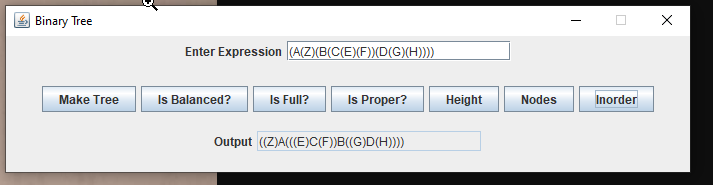
Height:



Nodes:

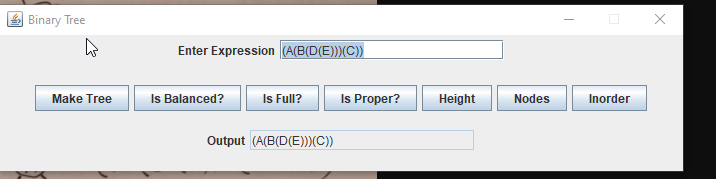


In Order:

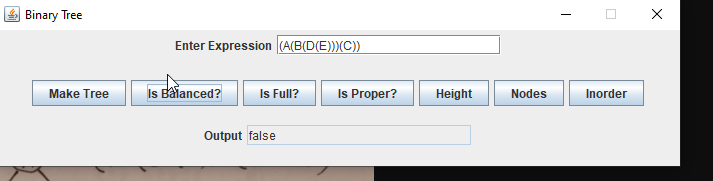


Test Case 3: (A(B(D(E)))(C))

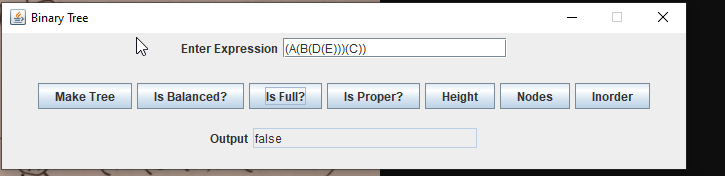
Make Tree



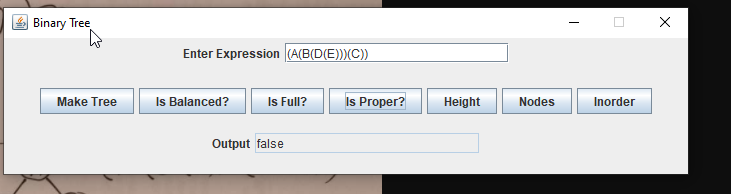
IS Balanced:



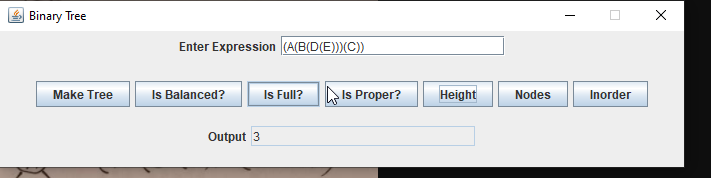
Is Full:



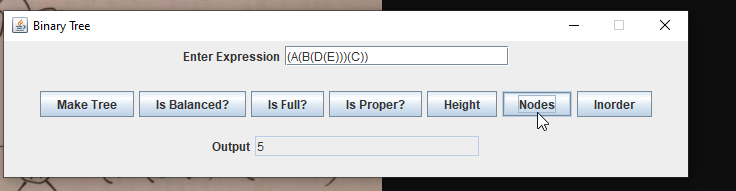
Is Proper:



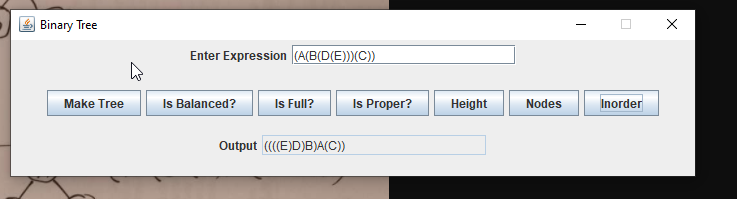
Height:



Nodes:



In Order:

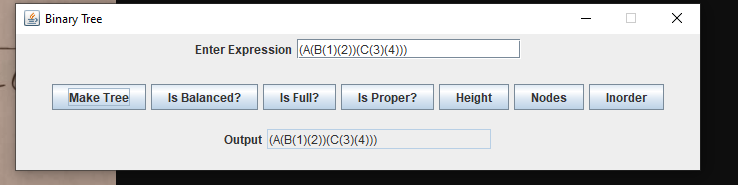


Lessons Learned:

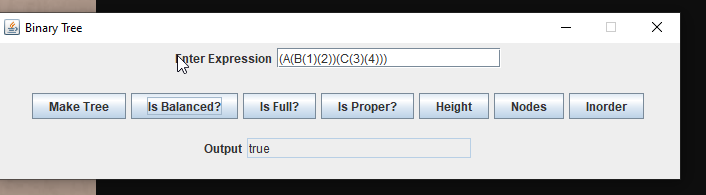
This was my favorite project so far. I haven’t done too much with recursion so far and I got a much better understanding of how to fully implement it into these type of programs. Working with the Binary Tree was interesting and how to work with the two separate branches. It made me understand better how the nodes would flow throughout the Tree and how to count them easily. This also helped me grasp of the different types of trees such as balance, full, and proper. I even researched into other ones like degenerate trees that are only one branch. I was tempted to put the functionally into this, but with being late on the assignments I just wanted to do the basics.

Test Case 4: (A(B(1)(2))(C(3)(4)))

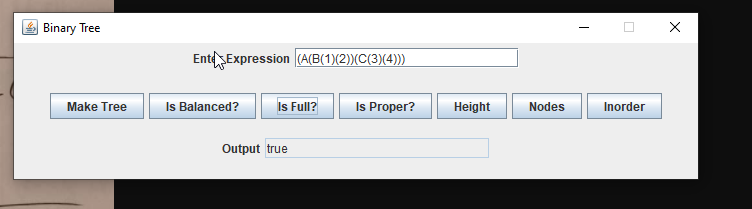
Make Tree:



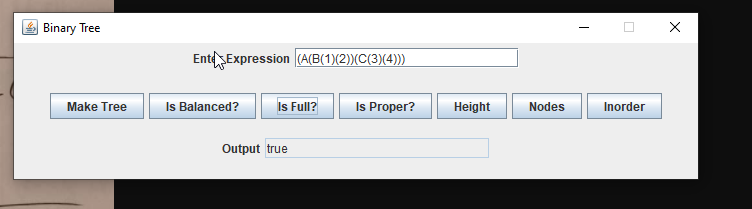
Is Balanced:



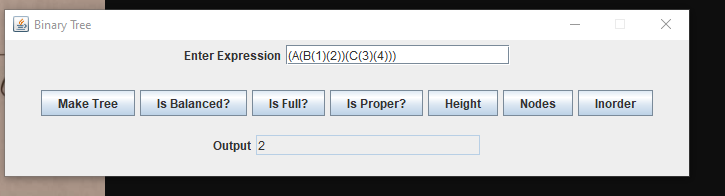
Is Full:



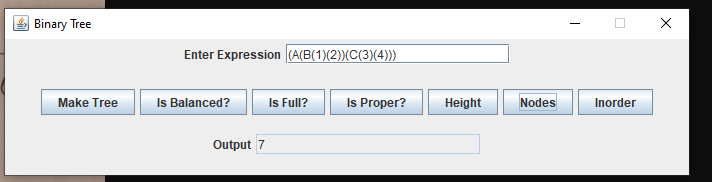
Is Proper:



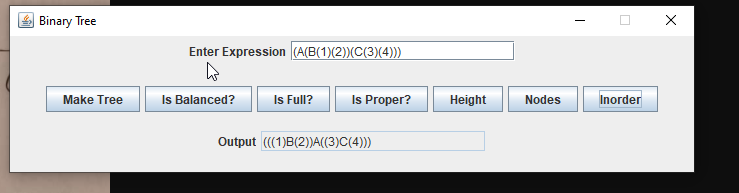
Height:



Nodes:



In Order:



No Input:

