Project 3 – Binary Search Trees

Burkhardt, Ryan

UMGC CMSC 315

Author Note

This program takes the users input and converts it to a binary search tree. It produces the proper output to show the user a binary search tree in the terminal with proper indented format. It also will tell the user if the binary search tree is balance or is a binary search tree.

UML Class Diagram

A screenshot of a computer

Description automatically generated

**Assignment 3**

For this Assignment we were given the task of creating a binary search tree with very specific input restrictions. This was where most of my problems came into place. I found it very easy to build a string builder that constructs the appropriate form of indentation, but I found recursion extremely difficult to obtain the grasp of. I essentially summed it up to a built in while loop but within itself. This in my mind works as a while there is more data the method such as binary tree will call itself until the binary tree is made. Either with a data as a node and a left and right expression. If there is no data to input, it will be entered as null.  
 The **Syntax Class** is a error catch class where you pass in the expression to run it through the error validation before the string is parsed into different expressions. It will catch all required syntax errors such as is binary tree, no digits, no null variables, etc. This uses the expression matches formatting learned from geeksforgeeks. Also, the digit \*=10 was extracted from geeksforgeeks to use in my code. This was to catch a digit character that was longer than one place.

The **Binary Tree Class** was my next class to take on. Here I started with the method of creating a tree which was to include three integers. How I made this was the data as the node and left and right as the children. This is also where I had the setter for the Binary Tree Object using the three integers or another Binary Tree object using the array list. In my array list I chose to use the mid method to find the middle of a sorted list and assign the data from there. This was in my opinion the easiest method because I knew my tree couldn’t get out of balance. The add data to list was my method to take the tree from the original input after it was displayed and make it into an array list. From this array list I would send it to the Binary Tree Array List to form the proper binary tree. After they would be sent to my main program in Project 3 to be displayed. Lastly, in the Binary Tree Class I had my getters for all three integers in the binary trees to include data, left and right.

The **Project3 Class** is my main class to include the main method. It includes the input to get from user to get a binary tree input. It runs the input through a series of methods to make sure it is in correct formatting as well as the proper length of parenthesis. This is where I had major confusion in how I was building my parse expression and that when I finished the parse expression it made much more sense. What the parse expression does is it counts the parenthesis until it knows there is a match of parenthesis, then it recursively runs this until there are only 3 parentheses left. Once it has the three parenthesis it takes the expressions and sends them to subexpressions to make them integers and plug into left and right. This was single-handedly my hardest process in the program due to lack of understanding. When you enter (1 2 3) this is not proper formatting of a binary tree this is simply a node with a value of 123. In my mind I believed that this should have been an unbalanced tree, and it should have corrected it to 2 1 3. But I realized in the assignment that the nodes must be separated by a parenthesis and that is when my project came together. So this became (1 (2 \* \*)(3 \* \*)). Once I realized that formatting was my error the problem became clearer.

In conclusion I will say this again is by far the hardest project I have faced. I found myself referring to the text every 5 minutes or on my next recursion question. I also had to get a tutor to go over recursion, it was a challenge to understand that concept. The Binary tree concept was also hard to understand on how to incrementally place the data and recursively. This was a challenge to find a parse method that works and sticks the proper data in the corresponding nodes. I learned a lot about the binary search tree, matches method, and recursion.

**Project 3 – Binary Search Trees**

**Test Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Input** | **What to Test** | **Actual Output** | **Pass/Fail** |
| 1 | (3(2\*\*)(5\*\*))  y  (3(2\*\*)(5\*\*)  n | Test the isBinary Tree and isBalanced method by giving it a simple balanced binary tree. Run without a closing parenthesis at the end to check for proper closing parenthesis on a double closer.Test while loop end by selecting n. | Displayed properly indented Binary Tree Found Error Program Closed Successfully | Pass |
| 2 | (3(2\*\*)(5\*\*)))  n | Check Syntax error catch on more LH parenthesis the RH | Caught Syntax Error and displayed error to user.  Program closed successfully. | Pass |
| 3 | (\*(\*\*\*)(\*\*\*))  n | Check to make sure it correctly identifies the no digits syntax error. | Correctly displayed no digits were present in the string. | Pass |
| 4 | (53 (28 (11 \* \*) (41 \* \*)) (83 (67 \* \*) \*))(63 (51 (20 (13 \* \*) \*) \*) \*)  (13 (53 \* \*) (11 (59 \* \*) \*)) | This will check every isbalanced and isbinary tree instance by providing it a balanaced binary tree, non-balanced binary tree, and a non-binary tree due to improper lh and rh nodes from our assignment Tests | Successfully displayed proper trees in every instance. Successfully closed all programs. | Pass |
| 5 | 0  Hi  Please  Come on  Fine  N | Testing to see if the expression is not properly formatted if it will catch it.  Testing our while loop for proper continue input | Successfully found it did not meet proper formatting.  Successfully prompted user for correct input. | Pass |

**Screenshots**

**Test Case 1**

**A screen shot of a computer

Description automatically generated**

**Test Case 2**

**A screen shot of a computer

Description automatically generated**

**Test Case 3**

**A screen shot of a computer

Description automatically generated**

**Test Case 4**

**A screenshot of a computer program

Description automatically generated**

**Test Case 5**

**A screenshot of a computer program

Description automatically generated**