**Binary Tree Infix Expression Parser**

Adam Carmichael

Evan Colyer

Lindsey Erwin

The program starts by reading a list of infix expressions from an input file. The input file is formatted so that each infix expression is on a single line. The program reads one line at a time from the input file. While it is reading over the input file it calls infixToPostfix() with the provided string to reformat it to postfix. Finally it creates a stack and calls evaluatePostfix() with the postfix string and the stack.

The first function is used to convert the infix expression from the input file into a postfix expression. The function creates two stacks, a tree, and a stringbuilder to create the postfix expression. One of the stacks is for tree nodes, and the other is for characters. It scans over the input line and pushes parenthesis to one of the stacks, then pushes the operators to the tree. It goes over the entire input expression, assigning children to the root. The function also checks the precedence of operators to make sure they are in the correct order so that the postfix expression can be calculated correctly. The precedence of operators is determined by another function that sets operators with the highest precedence with a higher integer number (The higher the precedence the higher the integer number). Once the entire infix expression has been parsed the function returns the postfix expression, and the second function is called to evaluate the postfix expression.

The second function, evaluatePostfix(), does post-order traversal of the tree and evaluates the expression by pushing the values in the tree to a stack. The function uses recursion to navigate through the tree, if it encounters an operand it pushes it to the stack, and if it encounters an operator it performs it on the two operands in the stack, and pushes the new result to the stack.

Two main ways that this program could be improved upon would be to fluidly parse a string without needing the format() function and to use a switch/case in the evaluateToPostfix() function.

|  |  |
| --- | --- |
| contributions |  |
| project report | Evan Colyer |
| Precedence function / main(driver) code | Evan Colyer |
| infixtopostfix function | Lindsey Erwin |
| evalualtepostfix function | Adam Carmichael |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **Description** | **Input** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| **1** | Check result of the infix expression parser | 1+2\*3 | 7 | 7 | Pass |
| **2** | Check result of the infix expression parser | 2+2^2\*3 | 14 | 14 | Pass |
| **3** | Check result of the infix expression parser | (1+2)\*3 | 9 | 9 | Pass |

