Assignment 7

CS329e - Elements of Software Design
Expression Tree
(100 points)
Due Date on Canvas and Gradescope

1 Description

For this assignment you will read from stdin the file expression.in and create an expression tree. The expression will be a valid infix expression with the all the necessary parentheses so that there is no ambiguity in the order of the expression. You will evaluate the expression and print the result. You will also write the prefix and postfix versions of the same expression without any parentheses.

In an expression tree the nodes are either operators or operands.

1. The operators will be in the set ['+', '-', '*', '/', '//', '%', '**'].
2. The operands will be either integers or floating point numbers.
3. All the operand nodes will be leaves of the expression tree.
4. All the operator nodes will have exactly two children.

Use the template program **ExpressionTree.py**, and you must follow the template exactly. You may add helper functions as needed.

The function *create_tree()* will take as input parameter an infix expression with parentheses as a String and create an Expression Tree from it. *Assume that the expression string is valid and there are spaces between the operators, operands, and the parentheses.*

You will take the expression string and break it into tokens. There are four different kinds of tokens - left parenthesis, right parenthesis, operator, and operand. When we read a left parenthesis we are starting a new expression and when we read a right parenthesis we are ending an expression. Here is the algorithm that you will use. Start with an empty node that is going to be your root node. Call it the current node. Then start parsing the expression.

1. If the current token is a left parenthesis add a new node as the left child of the current node. Push current node on the stack and make current node equal to the left child.
2. If the current token is an operator set the current node's data value to the operator. Push current node on the stack. Add a new node as the right child of the current node and make the current node equal to the right child.
3. If the current token is an operand, set the current node's data value to the operand and make the current node equal to the parent by popping the stack.
4. If the current token is a right parenthesis make the current node equal to the parent node by popping the stack if it is not empty.

Input:

Suppose the input file was the following:

```
( ( 8 + 3 ) * ( 7 - 2 ) )
```

Output:

For the input expression, this is what your program will output:

```
( ( 8 + 3 ) * ( 7 - 2 ) ) = 55.0

Prefix Expression: * + 8 3 - 7 2

Postfix Expression: 8 3 + 7 2 - *
```

The file that you will be submitting will be called **ExpressionTree.py**. We will be looking for good documentation, descriptive variable names, clean logical structure, and adherence to the coding conventions discussed in class.

References

Binary Expression Tree `https://en.wikipedia.org/wiki/Binary_expression_tree`

Pair Programming

For this assignment you may work with a partner. Both of you must read the paper on Pair Programming[1] and abide by the ground rules as stated in that paper. If you are working with a partner then only one of you will be submitting the code. But make sure that your partner's name and UT EID is in the header. If you are working alone then remove the partner's name and eid from the header.

1.1 Turnin

Turn in your assignment on time on Gradescope system on Canvas. For the due date of the assignments, please see the Gradescope and Canvas systems.

1.2 Academic Misconduct Regarding Programming

In a programming class like our class, there is sometimes a very fine line between ”cheating” and acceptable and beneficial interaction between students (In different assignment groups). Thus, it is very important that you fully understand what is and what is not allowed in terms of collaboration with your classmates. We want to be 100% precise, so that there can be no confusion.

The rule on collaboration and communication with your classmates is very simple: you cannot transmit or receive code from or to anyone in the class in any way – visually (by showing someone your code), electronically (by emailing, posting, or otherwise sending someone your code), verbally (by reading code to someone) or in any other way we have not yet imagined. Any other collaboration is acceptable.

[1]Read this paper about Pair Programming `https://collaboration.csc.ncsu.edu/laurie/Papers/Kindergarten.PDF`

The rule on collaboration and communication with people who are not your classmates (or your TAs or instructor) is also very simple: it is not allowed in any way, period. This disallows (for example) posting any questions of any nature to programming forums such as **StackOverflow**. As far as going to the web and using Google, we will apply the **”two line rule”**. Go to any web page you like and do any search that you like. But you cannot take more than two lines of code from an external resource and actually include it in your assignment in any form. Note that changing variable names or otherwise transforming or obfuscating code you found on the web does not render the ”two line rule” inapplicable. It is still a violation to obtain more than two lines of code from an external resource and turn it in, whatever you do to those two lines after you first obtain them.

Furthermore, you should cite your sources. Add a comment to your code that includes the URL(s) that you consulted when constructing your solution. This turns out to be very helpful when you’re looking at something you wrote a while ago and you need to remind yourself what you were thinking.

We will use the following Code plagiarism Detection Software to automatically detect plagiarism.

- Staford MOSS

  `https://theory.stanford.edu/~aiken/moss/`

- Jplag - Detecting Software Plagiarism

  `https://github.com/jplag/jplag` and `https://jplag.ipd.kit.edu/`