CS1527 Object-Oriented Programming

2021-2022

Expression Binary Tree

Important

This assessment is worth 30% of the overall marks for course CS1527.

You must **submit your code via Codio before 23.59 on 30th April**. Make sure your solution is ready to run before then.

The markers will look at your code, and will visually check its output. If the code is does not run, or the output is significantly different from that expected, you may get a much lower mark.

Remember that as this is an individual assessment, work submitted must be your own.

If you haven't already done so, you should familiarise yourself with the University guidance on plagiarism, available at https://www.abdn.ac.uk/sls/online-resources/avoiding-plagiarism/. Also note that, use of automated services to generate submissions for assessment will be treated as academic misconduct and pursued under the University misconduct procedures.

If you reuse other people's code or libraries, you must explicitly mention this in your code and acknowledge their work. This includes:

I. Where did you acquire them (e.g. URLs, books, GitHub)?

II. Who are the authors? They cannot be your classmates, of course!!

III. How did you modify them? In particular, if you make minor changes to a piece of other people's code, you must clearly mark in your code which part is your own code.

Introduction:

In Q1 of CS1527 Practical on Trees (Practical 8), you were given the following expression:

" (((5+2) *(2−1))/((2+9)+((7−2)−1)) *8) "

and were asked to convert it to a binary tree.

In this assessment you are asked to design a python program that can automatically convert a mathematical expression into a binary tree after the user enters it as an input. Specifically, the expression must be in the same format as the above, and a valid expression can be recursively defined as follows:

It must be in the form of (X?Y) where X and Y are either numbers or valid expressions, and ? stands for operators (*, /, +, -)

For instance,

(4*5) is a valid expression;
((2*3)*5) and ((2*4)*(5*6)) are also valid expressions;
(4*5*6) is not a valid expression because it has three operands within one pair of brackets;
((4*(5+6) is not a valid expression as the brackets are mismatched (final bracket missing).

To simplify the problem, we assume all numbers are single digit ones (i.e., ranging from 0 to 9) and all operators are: *, /, +, -

Hint: You are encouraged to use a combination of data structures, for instance, stacks and trees.

Tests

Writing test for your code is a good practice. Therefore, some unit tests should be included for your code to show its use, and to confirm that your data structures hold expected values.

Hint: Test the data structures, not output to the screen, as unit testing console output is challenging

Marking Criteria:

1. If your program runs, then that is 2 points, and if it runs but only does some parsing, then that is 4 points. If it both runs, and evaluates the input as a valid expression, then that is 6 points. [6pts]

   For example, given the input string: (((2*(3+2))+5)/2)

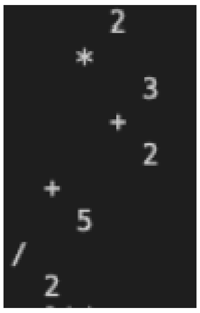
   The calculated result will be: 7.5

   Note that this example is for illustrative purposes, and during the marking process a different string will be used to test your solution.

2. Your program can visualise the generated binary tree with output in the terminal. If it has some buggy code for the traversal, then this is 3 points, and if it works fine, then this is 5 points. [5pts]

   The expected output tree for the input using an inorder traversal is as follows (you can print it on the screen, and you don't need to print the lines connecting the tree nodes if it is not convenient):

   You can "read" the tree if you imagine it tipped 90 degrees to the right. This makes " / " the root of the tree, and unlike the lectures, we read The tree from right to left.



```
      2
    *
        3
      +
        2
  +
    5
/
  2
```

3. Your program can output in the terminal the correct sequence of nodes of binary trees using preorder, postorder, and breadth first traversal. [5pts]

4. Your program can additionally report why the expression is not valid. If you have some reporting on in-valid expressions, then that is 3 points. If you have a more extensive list for reporting, such as below, then that is 5 points. [5pts]

(4*3*2)	Not a valid expression, wrong number of operands.
(4*(2))	Not a valid expression, wrong number of operands.
(4*(3+2)*(2+1))	Not a valid expression, wrong number of operands.
(2*4)*(3+2)	Not a valid expression, brackets mismatched.
((2+3)*(4*5)	Not a valid expression, brackets mismatched.
(2+5)*(4/(2+2)))	Not a valid expression, bracket mismatched.
(((2+3)*(4*5))+(1(2+3)))	Not a valid expression, operator missing.

5. You include some unit tests for your code. If there are working tests, and the tests match the criteria below, then that is 3 points. [3pts]

   Test names are explanatory, with explicit tests, and it makes sense that they are there.

6. Sensible coding rules are followed. [6pts]

   You should follow suitable coding conventions:
   - The code is easy to read; self-describing method names, which show intent, with short methods, and clear control flow.
   - The comments say why you are doing something and include references if appropriate.
   - Your code is maintainable and includes no dead code, which is never reached, or used, and is modularized in an understandable manner.
   - There is a 'readme' comment section at the top of the file explaining how to run the app (and any tests), plus any other relevant information.

Submission

Please make sure you use a single python file 'assessment2.py' to develop your solution. A solution that uses multiple python files will not be accepted. **You must submit your code for this assessment via Codio.** Make sure you upload your code to Codio before the deadline (Codio will save your work automatically so no need to 'Mark as Completed' as this option is greyed out), and test your code in Codio as well if you develop your work in other IDEs.

Final Notes:

Remember that as this is an individual assessment, work submitted must be your own. If you have not already done so, you should familiarise yourself with the University guidance on plagiarism, available at https://www.abdn.ac.uk/sls/online-resources/avoiding-plagiarism/

See the following guide for Python documentation:

http://docs.python-guide.org/en/latest/writing/documentation/