Expression trees parsing and execution of operations
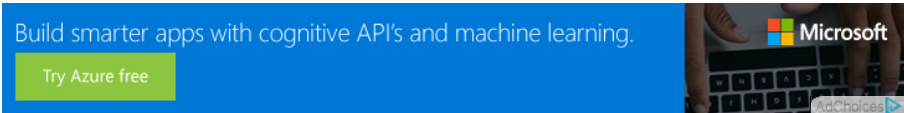
I got this problem in a coding challenge. I couldn't solve it on time but I still want to know how could it be done. I am not very familiar with expression trees and I found it hard to model the problem. The description goes like this:

Input: `expression_tree | sequence_of_operations`

The input is a single line of text with a expression tree and a sequence of operations separated by `|` character and ended by a `\n` newline character. Spaces are allowed in the input but should be ignored.

The expression tree is a sequence of 1-character variables A-Z and with sub expression trees formed by parenthesis (expression_tree). Examples: `AB`, `A(B C D)`, `(AB)C((DE)F)`

The sequence of operations is a string of with characters `R` (reverse) or `S` (simplify)

Reverse means reverse the order of everything in expression tree. Applying reverse twice in a row cancels out. Example: `(AB)C((DE)F) | R` should print `(F(ED))C(BA)`

Simplify means remove the parentheses around the very first element in the expression tree and each of its subexpression trees. Applying `S` multiple times should have same result as applying `S` once. Example: `(AB)C((DE)F) | S` should print `ABC(DEF)`

Output: Read the expression tree and apply the sequence of operations from left to right to the expression tree, print out the result without characters.

What I would like to know the most is how to model the expression tree to handle the parentheses and how does the simplify operation should work?

parsing   tree   expression

edited Nov 1 '16 at 3:09	asked Nov 1 '16 at 2:32
currarpickt	user3076480
1,654 3 12 28	**6** 3

Possibly: stackoverflow.com/questions/40102595/… – Nova Truong Dec 23 '16 at 19:41

1 Answer

```
'''
Examples are as follows :
INPUT:
(AB)(C(DE))/SS
(((AB)C)D)E/SS
(((AB)C)D)E/SR
(AB)C((DE)F)/SRS
(AB(CD((EF)G)H))(IJ)/SRS
(AB(CD((EF)G)H))(IJ)/SSSRRRS
(A(B(C)D)E(FGH(IJ)))/SRRSSRSSRRRSSRS
(A(BC(D(E((Z)K(L)))F))GH(IJK))/S
```