Assignment 4: Expression Trees

This assignment uses the following four files

`scanner.py`, `tree.py`, `expTree.py`, and `expTree_test.py`.

from Sections 2.4.5, 3.1.3, and 3.5.2 from the lecture notes. All files are available on Themis as `lab4expressiontrees.zip`.

Part 1 (3 points for tests)

Modify or extend the files such that infix (instead of prefix) expressions can be recognized and evaluated using expression trees. That is, the functionality of the functions `generate_expression_tree` and `_tree_node` in the file `expTree.py` needs to be adjusted so that it recognizes infix expressions and constructs the corresponding expression tree. *Hint: Note that these files already contain functionality that recognzes infix expressions, and functionality that generates expression trees.*

Upload all files of your program to Themis, except for the file `expTree_test.py`. This file will be added by Themis automatically.

An example of input and resulting output:

input	resulting **output**

`2 + 3 * 4 - 5 / 6`
`(2+3) * (4-5) /6`
`2 * x * x + 3 * x + 7`
`var1 - 3 * var2 /var3`
`+ 2 3`
`!`
	`give an expression: 2 + 3 * 4 - 5 / 6`
	`expression: ((2 + (3 * 4)) - (5 / 6))`
	`the value of this expression is: 13.166666666666666`
	
	`give an expression: (2+3) * (4-5) /6`
	`expression: (((2 + 3) * (4 - 5)) / 6)`
	`the value of this expression is: -0.8333333333333334`
	
	`give an expression: 2 * x * x + 3 * x + 7`
	`expression: ((((2 * x) * x) + (3 * x)) + 7)`
	`this is not a numerical infix expression`
	
	`give an expression: var1 - 3 * var2 /var3`
	`expression: (var1 - ((3 * var2) / var3))`
	`this is not a numerical infix expression`
	
	`give an expression: + 2 3`
	`this is not an infix expression`
	
	`give an expression: !`
	`good bye`

Part 2 (5 points for tests, 2 points for code review)

1. Define a function `simplify` that simplifies expression trees, according to the following requirements where E is an arbitrary expression:

   $0 * E$ and $E * 0$ are simplified to 0;
   $0 + E$, $E + 0$, $E - 0$, $1 * E$, $E * 1$ and $E/1$ are simplified to E.

2. Define a function `differentiate` that, given the simplified expression tree of an expression E, determines the expression tree of the derivative dE/dx. Use the function `simplify` to simplify the result of `differentiate`. For the computation of the

derivative, the following rules apply:

$$\begin{array}{lcll} dn/dx & = & 0 & \text{if } n \text{ is a number} \\ dy/dx & = & 0 & \text{if } y \text{ is an identifier different from } x \\ dx/dx & = & 1 & \\ d(E_1 + E_2)/dx & = & dE_1/dx + dE_2/dx & \\ d(E_1 - E_2)/dx & = & dE_1/dx - dE_2/dx & \\ d(E_1 * E_2)/dx & = & (dE_1/dx) * E_2 + E_1 * (dE_2/dx) & \\ d(E_1/E_2)/dx & = & ((dE_1/dx) * E_2 - E_1 * (dE_2/dx))/(E_2 * E_2) & \end{array}$$

Observe that, in the case of multiplication and division, both the original function and its derivative are involved. As a consequence, you will need a function to make a copy of an expression tree.

3. Extend the functionality of the program of Part 1 for non-numerical expressions with the functions `simplify` and `differentiate`.

Once again, upload all files of your program to Themis, except for the file `expTree_test.py`. This file will be added by Themis automatically.

An example of input and resulting output:

input	resulting **output**

`2 / (3 * 4) - (5 + 6)`
`2*x*x + 5*x - 6`
`0*x + 1*x*1*x*1 - 0`
`(x/1 - 2)/(x + 1)`
`2*y*y + 3*y - 7`
`!`
	`give an expression: 2 / ( 3 * 4 ) - ( 5 + 6 )`
	`the formula is ((2 / (3 * 4)) - (5 + 6))`
	`the formula evaluates to -10.833333`
	
	`give an expression: 2 * x * x + 5 * x - 6`
	`the formula is ((((2 * x) * x) + (5 * x)) - 6)`
	`this is not a numerical infix expression`
	`simplified: ((((2 * x) * x) + (5 * x)) - 6)`
	`derivative to x: (((2 * x) + (2 * x)) + 5)`
	
	`give an expression: 0 * x + 1 * x * 1 * x * 1 - 0`
	`the formula is (((0 * x) + ((((1 * x) * 1) * x) * 1)) - 0)`
	`this is not a numerical infix expression`
	`simplified: (x * x)`
	`derivative to x: (x + x)`
	
	`give an expression: ( x / 1 - 2 ) / ( x + 1 )`
	`the formula is (((x / 1) - 2) / (x + 1))`
	`this is not a numerical infix expression`
	`simplified: ((x - 2) / (x + 1))`
	`derivative to x: (((x + 1) - (x - 2)) / ((x + 1) * (x + 1)))`
	
	`give an expression: 2 * y * y + 3 * y - 7`
	`the formula is ((((2 * y) * y) + (3 * y)) - 7)`
	`this is not a numerical infix expression`
	`simplified: ((((2 * y) * y) + (3 * y)) - 7)`
	`derivative to x: 0`
	
	`give an expression: good bye`