Updated on March 9th 2009, Monday 22:30

CENG 242

Homework #1
(Due: March 17th 2009, Tuesday 23:55)

You are given a parse tree of an arithmetic expression and asked to generate the infix expression as the result. Assume that tree is defined as:

data Tree = Empty | Leaf Char | Branch Char Tree Tree deriving Show

You will write a function **treetoinfix** in the form:

treetoinfix <tree>

where *tree* is the parse tree. The function should return the infix expression. Apply proper parenthesization. DO NOT include any unnecessary parentheses.

Assume that only these arithmetic operators will be used in the expression: + , - , * , / .
Also consider the case of unary minus.

UPDATE: Assume that the only unary operator is unary minus (-).

Example:

treetoinfix (Branch '*' (Branch '+' (Leaf 'a') (Leaf 'b')) (Leaf 'c'))
"(a+b)*c"

treetoinfix (Branch '+' (Branch '+' (Leaf 'a') (Leaf 'b')) (Leaf 'c'))
“a+b+c”

treetoinfix (Branch '*' (Branch '/' (Leaf 'a') (Leaf 'b')) (Branch '/' (Leaf 'c') (Leaf 'd')))
"a/b*c/d"

treetoinfix (Branch '+' (Branch '-' (Leaf 'a') (Branch '*' (Leaf 'b') (Leaf 'c'))) (Branch '/' (Leaf 'd') (Leaf 'e')))
"a-b*c+d/e"

treetoinfix (Branch '-' Empty (Leaf '3'))
"-3"

treetoinfix (Branch '+' (Leaf '5') (Branch '-' Empty (Leaf '3')))
"5+-3"

Updated on March 9th 2009, Monday 22:30

treetoinfix (Branch '-' (Leaf '5') (Branch '-' Empty (Leaf '3')))
"5--3"

UPDATED (clarifying unary minus): treetoinfix (Branch '*' (Leaf '5') (Branch '-' Empty (Leaf '3')))
"5*-3"

UPDATE (additional examples):

treetoinfix (Branch '/' (Branch '/' (Leaf 'a') (Leaf 'b')) (Branch '/' (Leaf 'c') (Leaf 'd')))
"a/b/(c/d)"

treetoinfix (Branch '/' (Branch '*' (Leaf 'a') (Leaf 'b')) (Branch '/' (Leaf 'c') (Leaf 'd')))
"a*b/(c/d)"

treetoinfix (Branch '/' (Branch '/' (Leaf 'a') (Leaf 'b')) (Branch '*' (Leaf 'c') (Leaf 'd')))
"a/b/(c*d)"

treetoinfix (Branch '/' (Branch '*' (Leaf 'a') (Leaf 'b')) (Branch '*' (Leaf 'c') (Leaf 'd')))
"a*b/(c*d)"

treetoinfix (Branch '*' (Branch '/' (Leaf 'a') (Leaf 'b')) (Branch '/' (Leaf 'c') (Leaf 'd')))
"a/b*c/d"

treetoinfix (Branch '*' (Branch '*' (Leaf 'a') (Leaf 'b')) (Branch '/' (Leaf 'c') (Leaf 'd')))
"a*b*c/d"

treetoinfix (Branch '*' (Branch '/' (Leaf 'a') (Leaf 'b')) (Branch '*' (Leaf 'c') (Leaf 'd')))
"a/b*c*d"

treetoinfix (Branch '*' (Branch '*' (Leaf 'a') (Leaf 'b')) (Branch '*' (Leaf 'c') (Leaf 'd')))
"a*b*c*d"

treetoinfix (Branch '/' (Branch '*' (Branch '/' (Leaf 'a') (Leaf 'b')) (Leaf 'c')) (Branch '/' (Branch '*' (Leaf 'k') (Leaf 'm')) (Leaf 'l')))
"a/b*c/(k*m/l)"

treetoinfix (Branch '/' (Branch '*' (Branch '-' (Branch '/' (Leaf 'a') (Leaf 'b')) (Leaf 'j')) (Leaf 'c')) (Branch '/' (Branch '-' (Branch '+' (Leaf 'k') (Leaf 'm')) (Leaf 'x')) (Branch '-' (Leaf 'l') (Branch '*' (Leaf 't') (Leaf 'z')))))
"(a/b-j)*c/((k+m-x)/(l-t*z))"

treetoinfix (Branch '*' (Branch '/' (Leaf 'a') (Leaf 'b')) (Branch '/' ((Branch '/' (Leaf 'c') (Leaf 'd'))) ((Branch '/' (Leaf 'e') (Leaf 'f')))))
"a/b*c/d/(e/f)"

Updated on March 9th 2009, Monday 22:30

UPDATE (clarifying unary minus): treetoinfix (Branch '+' (Branch '/' (Leaf 'a') (Leaf 'b')) (Branch '/' (Branch '-' Empty (Leaf 'c')) (Leaf 'd')))
"a/b+-c/d"

Specifications:

- All the work should be done done **individually**.
- Your codes should be written in Haskell and have the name "**hw1.hs**"
- In evaluation, black box method will be used. So be careful about the name of functions, data structures etc.
- You will submit your code through **Cow** system.
- You should test your codes in **inek** machines with **hugs** before submitting.