

Wednesday, March 9, 2016

**DUE:**  
**Purpose:**  
**Problem:**  
**Method:**

To solve a non-trivial programming problem using binary trees.

For this assignment you are to implement three non-member functions to

- 1) determine if a tree contains only operators and constants (and no variables),
- 2) evaluate a binary expression tree (if it only has operators and constants) and
- 3) reduce a binary expression trees.

You will also modify the inorder traversal function to print out fully-parenthesized infix expressions.

We can apply several reductions to a binary expression tree and arrive at a reduced tree that represents an expression equivalent to the original one. Some of the reduction rules are

$a + 0$	$\rightarrow$	$a$
$a - 0$	$\rightarrow$	$a$
$a * 0$	$\rightarrow$	$0$
$a / 1$	$\rightarrow$	$a$
$0 + a$	$\rightarrow$	$a$
$0 * a$	$\rightarrow$	$0$
$1 * a$	$\rightarrow$	$a$

constant operator constant  $\rightarrow$  constant (ex.  $4 + 4 \rightarrow 8$ )  
where a is an arbitrary variable.

For example, applying the reduction rules given above to the binary expression tree given in figure 1 results in the binary expression tree in figure 2.

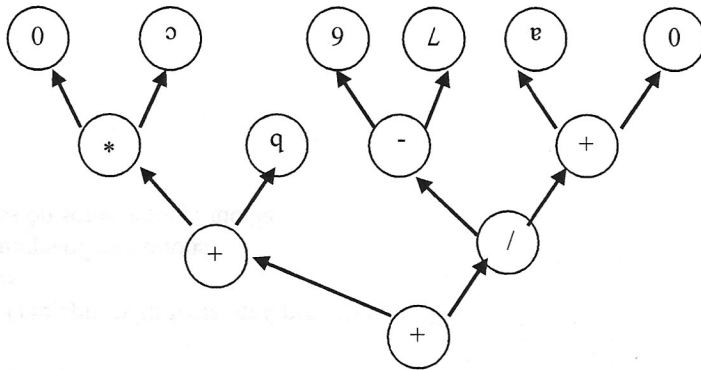


figure 1

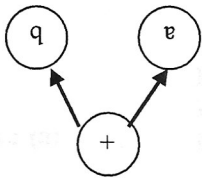


figure 2

## Binary Expression Tree:

A binary expression tree is a special case of a binary tree, in which all the interior nodes have exactly two children. The info field of an interior node of a binary expression tree contains an