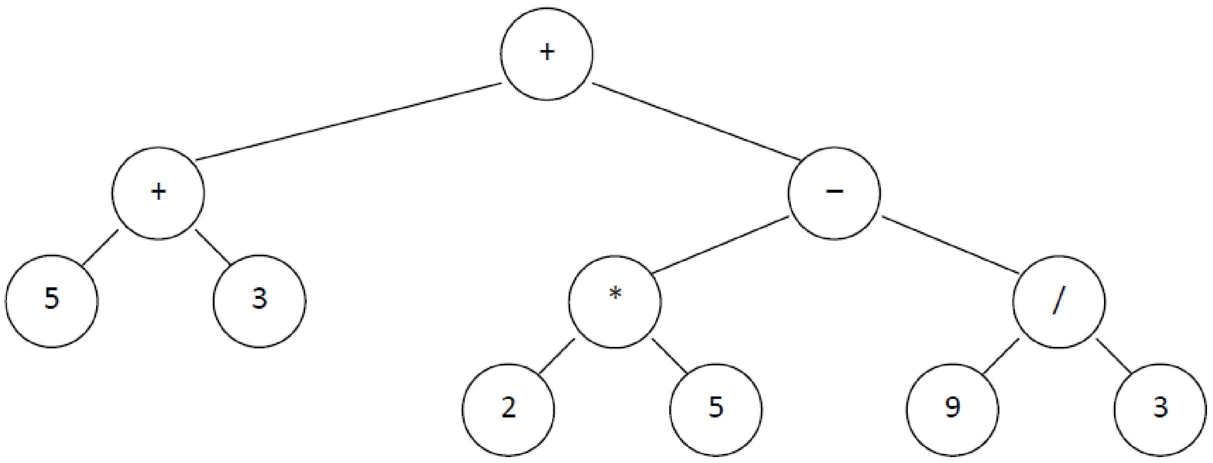
## COEN 70: Formal Specification and Advanced Data Structures

### Winter 2015

### Lab 8: Expression Tree

We are going to implement the algebraic binary expression tree (`ExpTree`). This is a binary tree where each non-leaf nodes contain an operator and the leaf nodes contain an operand. Below is an example of such a tree for the expression:



We will limit the capabilities to integer operands and to the four basic arithmetic operators (+,-,*,/). Because the node can either have an operator or an operand, we will be creating an `Expression` object that has an `int` variable for the operand and a `character` variable for the operator. Simply create a "public" class (data members are `public`). A node data such as this can also help in the evaluation process where the "operator" node can store the temporary result of the operation. However, in outputting the tree, these temporary values should not be printed.
Beside the basic methods, our `ExpTree` class also has the following:

- `build()` – create the tree based on a postfix expression (recall Lab4) given by the user via the keyboard
- `evaluate()` – recursively evaluate every node and output the value at the root
- `print_preorder()` – output the tree with pre-order traversal
- `print_inorder()` – output the tree with in-order traversal
- `print_postorder()` – output the tree with post-order traversal

The `print_postorder()` output should match with the input string.
You are to implement this `ExpTree` class and an appropriate test program.