Arithmetic Expression Trees (Part 2)
Cpt S 321 Homework Assignment
Washington State University

Submission Instructions: (see syllabus)

Assignment Instructions:

Read each step's instructions *carefully* before you write any code.

In this assignment you will finish what you started in the previous homework assignment by implementing an arithmetic expression parser that builds a tree for an expression. This tree can then be used for evaluation of the expression. You may want to refer back to the instructions for the first half of this assignment to make sure that everything was properly implemented there and that you are following the required naming conventions.

Recall that your expression tree class is implemented in your engine DLL and demoed in this assignment through a standalone console application that references the DLL. Integration into the spreadsheet will happen in a future homework assignment.

Requirement Details:

Support parentheses and operators with proper precedence (3 points):

- Support the addition and subtraction operators: + and -
  - You may assume that all instances of the minus character are for subtraction. In other words, you don't have to support negation.
- Support the multiplication and division operators: * and /
- Each operator must be implemented with proper precedence.
- Parentheses must be supported and obeyed. Implementations without support for parentheses will result in a 50% deduction on all working operators. For example, if you support all 4 operators correctly but not parentheses, you get 1.5 / 3 points for this part.

Tree Construction (4 points):

- Build the expression tree correctly internally. Non-expression-tree-based implementations will be penalized 4 points.

Support for Variables (2 points):

- Support correct functionality of variables including multi-character values (like "A2").
- Variables will start with an alphabet character, upper or lower-case, and be followed by any number of alphabet characters and numerical digits (0-9).

- As you build the tree, every time you encounter a new variable add it to the dictionary of variable values with a default value of 0. This will be needed for integration with the spreadsheet application in a future assignment.
- Variables are stored per-expression, so creating a new expression should clear out the previous set of variable values. However, all variable values must persist as long as the expression is NOT being changed.
    - Ex: You shouldn't reset any variables after an evaluation or anything like that

<u>Clean, REUSABLE code (1 point):</u>

- Have clean, well-documented code and an easy to use interface for grading.
- Obey all the naming requirements outlined in the first part of this homework (as well as any added here obviously)

10 points total