Arithmetic Expression Trees (Part 1 of 2)
Cpt S 322 Homework Assignment #5
by Evan Olds

Submission Instructions:

Submit source code (zipped) to Angel BEFORE the due date/time. If the Angel submission is not working, then submit to TA via email BEFORE the due date/time. "Angel wasn't working" is never an excuse.

Optional: Include a readme.txt file in the zip with any relevant information that you want the grader to be aware of.

Assignment Instructions:

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

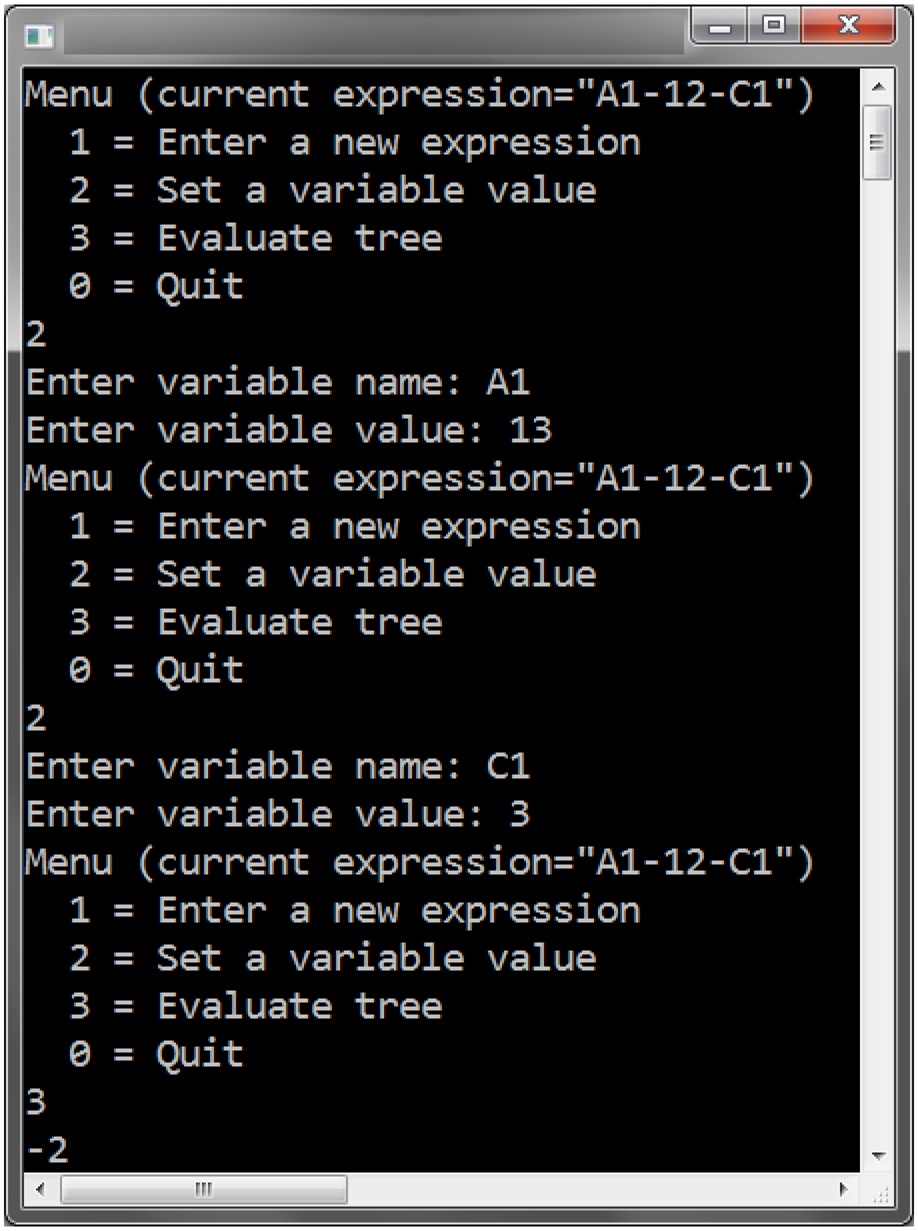
Note: This is a standalone console application and integration into the spreadsheet will happen in a future homework assignment. Make sure you implement the code in a class or collection of classes that can be easily moved to another application later on.

In this assignment you will implement an arithmetic expression parser that builds a tree for the expression. This tree can then be used for evaluation of the expression. The parsing aspect of this assignment is simplified and you will extend the parser in the next homework. But the entire evaluation functionality, including setting variable values, will be implemented in this assignment.

Create a console application that presents a menu when run. This menu must contain the following options:

1. The option to enter an expression string. You may assume that only valid expressions will be entered with no whitespace during grading. Simplified expressions are used for this assignment and the assumptions you can make are discussed later on.
2. The option to set a variable value in the expression. This must prompt for both the variable name and then the variable value.
3. The option to evaluate to the expression to a numerical value.

The screenshot below shows what this might look like:



```
Menu (current expression="A1-12-C1")
  1 = Enter a new expression
  2 = Set a variable value
  3 = Evaluate tree
  0 = Quit
2
Enter variable name: A1
Enter variable value: 13
Menu (current expression="A1-12-C1")
  1 = Enter a new expression
  2 = Set a variable value
  3 = Evaluate tree
  0 = Quit
2
Enter variable name: C1
Enter variable value: 3
Menu (current expression="A1-12-C1")
  1 = Enter a new expression
  2 = Set a variable value
  3 = Evaluate tree
  0 = Quit
3
-2
```

Requirement Details (10 points total):

Support simplified expressions (2 points):

- For this assignment all expressions will be simplified next to what you'll have to support in the next homework. As stated earlier, this assignment is primarily about getting the correct evaluation logic in place for the tree.
- Assume expressions will NOT have any parentheses
- Assume expressions will only have a single *type* of operator, but can have any number of instances of that operator in the expression
    - Example 1: expression could be "A+B+C1+Hello+6"
    - Example 2: expression could be "C2-9-B2-27"
    - Example 3: expression could NOT be "X+Y-Z" because that has two different types of operators: + and –
- (In the next assignment you'll have to support all valid arithmetic expressions)
- Support operators +,-,*, and / for addition, subtraction, multiplication, and division, respectively. Again, only one type will be present in an expression that the user enters.
- Parse the expression that the user enters and build the appropriate tree in memory

Tree Construction and Evaluation (5 points):

- Build the expression tree correctly internally. Non-expression-tree-based implementations will not be worth any points.
- Each node in the tree will be in one of three categories:
    - Node representing a constant numerical value
    - Node representing a variable
    - Node representing a binary operator
- Your expression class must have a public evaluation function that takes no parameters and returns the value of the expression as a double.
    - public double Evaluate()

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).
- A set of variables is stored per-expression, so creating a new expression will clear out the old set of variables.
- Have a default expression, something like "A1+B1+C1" would be fine, so that if setting variables is the first action that the user chooses then you have an expression object to work with.

Clean, REUSABLE code (1 point):

- As we discussed in class you can have this functionality in a class called “Expression”, “ExpTree”, “Exp”, or something else similar. You can name it whatever you want, but make sure all the relevant functionality is in this class. Do <u>not</u> have things like the variable dictionary declared as a local variable in Main(). Such implementations will not allow you to easily bring your code into the WinForms app for the next homework.
- Have clean, well-documented code and an easy to use interface for grading.