This lab concerns binary trees.

1. Suppose that we restrict ourselves to arithmetic expressions consisting of single-digit integers (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), parentheses ((,)), and the usual arithmetic operators (+, -, *, /). We also allow negation, e.g. -5, so that the minus sign is being used as both a binary and unary operator. Consider a typical infix mathematical expression, e.g. \((2*3)+(6*7*(-5))/8\). It is possible to associate a binary parse tree whose non-leaf nodes are operators +, -, *, / and whose leaf nodes are single-digit numbers. In the case of the example above, we can make the root node + and have the left sub-tree describe the computation of \(2*3\) while the right sub-tree describes the computation of \((6*7*(-5))/8\).

2. It is possible to write a program which will parse and evaluate an algebraic expression as described in Part 1. Get the sample program calc.cpp, compile and link it, and try it out on the expression above, for example:
   ```plaintext
   echo "(2*3)+(6*7*(-5))/8" | calc -d
   ```
   The “-d” flag turns on debugging and lets you see how the expression is being evaluated recursively with calls to the three procedures: `expr()`, `term()`, and `factor()`. If you omit it the program runs silently and just gives you the final (integer) value. The comment at the beginning of the program calc.cpp explains how an expression is parsed into terms and factors.

3. It is also possible to convert an infix expression such as \((2*3)+(6*7*(-5))/8\) into postfix (also called RPN, as used in some HP calculators) such as \(23*67*5-*8+/\). Note that the \(-\) in the previous postfix expression is a “negation” or “change sign (CHS)” and not a subtraction. This is why it is better to have two different buttons - and CHS as on an HP calculator.

**Assignment** The assignment is to turn in your answers to each of the following two problems. Email me your results as in-line text. Please:

   i. Please keep line length under 72 characters per line.

   ii. Please do not send me any attachments.

1. Suppose that you built a binary parse tree for an expression similar (but not limited) to the above whose non-leaf nodes are operators (+, -, CHS, *, /) and whose leaf nodes are single-digit integers. What type of traversal would generate the postfix sequence of numbers and operators? Give at least two non-trivial examples of expressions and the traversals. You can write the trees in ascii text, e.g.
   ```plaintext
   +
   /  \
   *  8
   /  \
   2  3
   ```

2. Given a binary tree describing the expression (in some graph-data format such as we used for the program dacgraph), how could you find the value of the expression using a stack machine which would first find a starting node and then do traversal with evaluations of values at carefully chosen times? Give me pseudo-code which would do the proper things and which is specific enough for me to be able to code a program in C/C++.