CS242: Object-Oriented Design and Programming

Programming Assignment 5
Part 1 due Wednesday, November 20th, 1995
Part 2 due Tuesday, November 28th, 1995
Part 3 due Tuesday, December 4th, 1995

Problem Statement

In this assignment you will write an operator-precedence parser. The parser will construct a syntax tree for each input line and then evaluate it. You will be parsing a language that is a subset of C expressions.[1] The underlying grammar, illustrating the operator precedence, is succinctly stated as:

```
start        ::= assign_expr
assign_expr  ::= add_expr | ID assign_op assign_expr
add_expr     ::= mult_expr | add_expr add_op mult_expr
mult_expr    ::= unary_expr | mult_expr mult_op unary_expr
unary_expr   ::= primary | uminus_op primary
primary      ::= ID | NUM | l_paren assign_expr r_paren
add_op       ::= + | -
mult_op      ::= * | /
uminus_op    ::= -
assign_op    ::= =
l_paren      ::= (
r_paren      ::= )
```

Your program will be developed in the following three parts:

1. *Lexical analysis* – Write a lexical analyzer that reads test input and "tokenizes" it (*i.e.,* returns an appropriate `enum` and associated value for each type of token it reads.
2. *Parsing and expression tree construction* – Write an operator precedence parser that will parse the tokens and build an expression tree. A rough sketch of the operator precedence parsing algorithm will be presented in the class slides. More information is available in the Aho, Sethi, and Ullman book on compilers.
3. *Expression tree traversal* – Implement "in order," "pre order," "post order," and "level order." traversals of the syntax tree. In addition, implement a function that evaluates the "yield" of the tree and prints it out to stdout (just like the sample program I gave you).

`/project/adaptive/cs242/assignment5` contains sample test input (`testinput`) and a working sample parser (`opp`). You should run the test program to see how your program's output should appear. It is very important that your output match this form.

[1]Note that the current implementation only handles one letter, lower-case variable names.