

CSCE 2050
Programming Assignment 5
Fall 2006
Due Date: Friday, Dec. 8

Requirements [Lists, Stacks, Symbol Table, and Trees.] The difference between this assignment and the last is in the specifications section. Read from the input stream a sequence of (infix) arithmetic statements composed of

- Nonnegative integers: 0, 1, 2, etc.
- Variables: a, b, c, ..., z
- Parentheses: (and)
- Arithmetic operators: +, -, *, /, ^, and =

A semicolon will be used to denote the end of each statement.

Statements will be in the form of assignment statements, that is, a variable (the left variable) followed by '=' followed by an arithmetic expression. Translate the statement to an internal code, then evaluate the statement using the postfix. The evaluation of a subexpression consisting in part of a left variable will be to update the value of the left variable in the symbol table. When the end of the list of statements is encountered (i.e., a statement consisting only of a semi-colon) print the values of each variable used in the the sequence of statements.

Specifications The input should appear as follows:

```
Please enter a sequence of arithmetic statements
a = 2;
b = 4^a - 6;
c = 3*a + b;
;
```

The output should appear as follows:

Variable	Value
a	2
b	10
c	16

The symbols must appear in alphabetical order in the output list. Please note also the following issues that your code must take into account.

- If the expression does not have balanced parentheses, an error message should be printed, the stack(s) should be made empty, and the program proceed to the next statement.
- If a statement does not have a left variable, do the same.
- Note that variables may have negative values as via the statement: $a = 0 - 2$.
- All arithmetic is integer arithmetic so 2^a would be zero for all negative values of a. You do not need to check for this. It is a natural outcome of integer arithmetic.
- Note that it is possible for a variable to appear to the right of equal (=) that has not been assigned a value. This is also an error.

A `C++` class must be used to maintain stacks. Appropriate member operators must be provided so the stacks can be manipulated. It is recommended but not required that yet another class be used to maintain lists. You may use the standard library functions if you prefer.

There must be a symbol table class! The symbol table class must be organized as a *binary search tree*. You must have a method that executes an in-order traversal. This method is used to print the contents of the symbol table in alphabetical order.

Design and Testing No specific test case is prescribed. Run more than one test case. Your test cases must include checking for all the errors (balanced parentheses, etc.) noted in the list above.

Error Handling Error checking must at least include assuring the next input symbol is a legitimate token, that the parentheses of the expression are in balance, that the symbols used on the right of `=` have been assigned values, and that there is a left variable.

Implementation Due Date Friday, 8 December. Submit the following using the *project* command. Select the correct section of the class, 2050s001. The project id is **phw4**.

- The source code (i.e., the `*.h` and `*.cpp` files)
- Either of the following
 - A makefile that compiles the code and executes the test cases
 - A text file created using the “script” command that shows execution of the test cases