For this assignment we are expanding on assignment 3. Now that we have parsed input, we will execute the code.

As a reminder, the language has the following rules:

1. An empty string is a semantic error.
2. The language contains only two types: a string and an integer.
3. The language contains only two statements: a print statement, and a set statement.
4. PRINT means evaluate the expression and printing the result on standard out.
5. SET means evaluate the expression and save the value in the variable named ID
6. It is an error to use a variable that has not been set.
7. The addition and multiplication operators associate left-to-right.
8. Addition is defined only between two integers or two strings. String addition is concatenation.
9. Multiplication is defined between two integers or between an integer and a string. Multiplying a string by an integer X repeats the string X times.
10. The language includes a subscripting operator which may have one or two expressions inside the square bracket, separated by a semicolon. Both expressions must evaluate to integers, which are used as indexes into the string (starting, of course, from zero). A single expression results in a string containing the single character at that position. Two expressions results in a substring beginning at the first position and ending at the character before the second position. If either of the expressions would access a character that is not in the string, this will be deemed a runtime error.
11. All other combination of types and operations are undefined.

For this assignment, after parsing you must perform all semantic checks and execute the program. Please note that some of the checks can be performed statically (an empty string constant is an error, and using a variable before setting it is an error), but some can only be performed at run time.

Failing a static check should print an error message and stop execution. Failing a dynamic check should print the string RUNTIME ERROR followed by a description of the error (for example; RUNTIME ERROR types being added are not permitted).

Observe that doing this assignment involves evaluating the parse tree, which is a recursive operation. Defining a virtual eval() function, with different implementations in different classes, is a useful approach. It's also likely that you will need to define some idea of a Type and a Value for this assignment. The eval() function will likely want to return a Value, and will need to test a Type.

Your program should read the file whose name is passed as a command line argument, or the standard input if no command line argument is provided. You may divide this assignment into as many files as you like. You MUST use p2lex.h from the last assignment, with no changes. You MUST have your lexical analyzer in a separate file.