CS 280: Program #3

Fall 2016

Due November 22, by 11:55pm, via Moodle

For this assignment we are expanding on assignment 2. We will create a parser for a small language that uses the tokens that we recognized using our assignment 2 parser. You will be provided with an assignment 2 parser, which you may use if you do not want to use your own.

The grammar for our language is:

```
Program ::=    StmtList
StmtList ::=   Stmt | Stmt StmtList
Stmt ::=  PRINT Expr SC | Set ID Expr SC
Expr ::=  Expr PLUS Term | Term
Term ::=  Term STAR Primary | Primary
Primary ::=    ID | String | INT | LPAREN Expr RPAREN
String ::= STR | STR LEFTSQ Expr RIGHTSQ | STR LEFTSQ Expr SC Expr RIGHTSQ
```

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.

You must write a recursive descent parser to parse the grammar, one C++ function per rule in the grammar. During the parse, you should print error messages if any syntax errors are detected.

If your parse is successful (that is, if it detects no errors) the result of the parse will be a parse tree. If your program successfully generates a parse tree, then the program should do the following:

1. Traverse the tree and print a count of the number of each of the operands (+, *, []) in the tree
2. Traverse the tree and make sure rule #1 and #6 are always followed; print an error if it is not

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.