CS 280: Program #3

Spring 2016

Due April 11, 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 ::=  PRINTKW Aop SC | INTKW VAR SC | STRKW VAR SC | Aop SC
Aop ::=   VAR EQOP Aop | Expr
Expr ::=  Expr PLUSOP Term | Expr MINUSOP Term | Term
Term ::=  Term STAROP Primary | Primary
Primary ::=    SCONST | ICONST | VAR | LPAREN Expr RPAREN
```

The language has the following rules:

1. An empty string should be treated as a semantic error.
2. The language contains only two types: a string (a string constant is an SCONST) and an integer (an integer constant is an ICONST).
3. The language contains only four statements: two variable declarations, a print statement, and an assignment statement.
4. PRINT means evaluating the expression and printing the result on standard out.
5. It is an error to use a variable that has not been declared.
6. The addition, subtraction and multiplication operators associate left-to-right; assignment associates right to left.
7. Addition is defined only between two integers or two strings. String addition is concatenation.
8. Subtraction is only defined between two integers
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. Assignment is only permitted between like types: integer to integer or string to string.
11. The value of an assignment expression is the value that is assigned.
12. 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 SCONST, ICONST and VAR in the tree
2. Traverse the tree and print a count of the number of each of the operands in the tree
3. Traverse the tree and make sure rule #1 and #5 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.