NJIT New Jersey's Science & Technology University	**CS 280: Program #3** Spring 2015 Due April 10 (note new date), by 11:55pm, via Moodle

For this assignment we are expanding on assignment 2. We will create a parser for a language that uses the tokens that we recognized using our assignment 2 parser.

The grammar for our language is

```
Program ::=    StmtList
StmtList ::=   Stmt | Stmt StmtList
Stmt ::=  PRINT Expr SC | SET ID Expr SC
Expr ::=  Expr PLUS Term | Expr MINUS Term | Term
Term ::=  Term STAR Primary | Term SLASH Primary | Primary
Primary ::=    ID | INT | STRING
```

The language has the following additional semantic rules:

1. The language contains two types: integer (an INT token) and string (a STRING token)
2. Printing means evaluating the expression and printing the result on standard out
3. Setting means evaluating the expression and assigning its type and value to the ID
4. It is an error to use a variable in an Expr if it has not been set
5. All four operators work on pairs of INTs; the resulting type of such an operation is an INT
6. Addition of strings concatenates the strings and results in a string
7. Subtracting two strings results in a string where the second string is removed from the first string if it is present; if it is not present, the result is the first string
8. Multiplying a string by an integer results in a new string where the string operand is repeated "integer" times
9. Dividing a string by an integer results in a substring of up to the first "integer" characters
10. Dividing an integer by a string results in a substring of up to the last "integer" characters
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 make sure semantic rule #4 is always followed; print an error message in all places where it is now
2. Traverse the tree and count the leaves

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.