Compilers

Project One

Write a complete lexer and parser that validates the source code and creates a symbol table (including type data) for the following grammar.

```
Program       ::== Statement $

Statement     ::== P ( Expr )
              ::== Id = Expr
              ::== VarDecl
              ::== { StatementList }

StatementList ::== Statement StatementList
              ::== ε

Expr          ::== IntExpr
              ::== CharExpr
              ::== Id

IntExpr       ::== digit op Expr
              ::== digit

CharExpr      ::== " CharList "

CharList      ::== Char CharList
              ::== ε

VarDecl       ::== Type Id

Type          ::== int | char

Id            ::== Char

Char          ::== a | b | c ... z

digit         ::== 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

op            ::== + | -
```

Notes and Additional Requirements:

- The lexer is not as simple as our examples in class, so be careful.
- Provide both errors and warnings. Warnings are non-fatal mistakes or omissions that your compiler can and will correct. Forgetting the `$` after `Statement` is one example. If this happens output a warning and then add the missing symbol so that compilation continues.
- When you detect an error report it in helpful detail including where it was found.
- Include optional verbose output functionality that traces the stages of the parser including the construction of the symbol table. (An option for *GLaDOS mode* is okay, but better *Yoda mode* is. *Horror movie mode* might be fun too.)

Create several test programs that cause as many different types of errors as you can in order to thoroughly test your code. (Think about code coverage). Include several test cases that show it working as well. Write up your testing results in a document to hand in at the beginning of the class in which this is due.

E-mail me all of your files for this project before the beginning of the class in which this is due.