Test Case # 1:

{}\$		

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...

DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1

DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 1

DEBUG Lexer - EOP [ $ ] found on line 1

LEXER: Lex completed with 0 error(s)

PARSER: Parsing program 1...

PARSER: parse() called

PARSER: parseProgram()

PARSER: parseProgram()

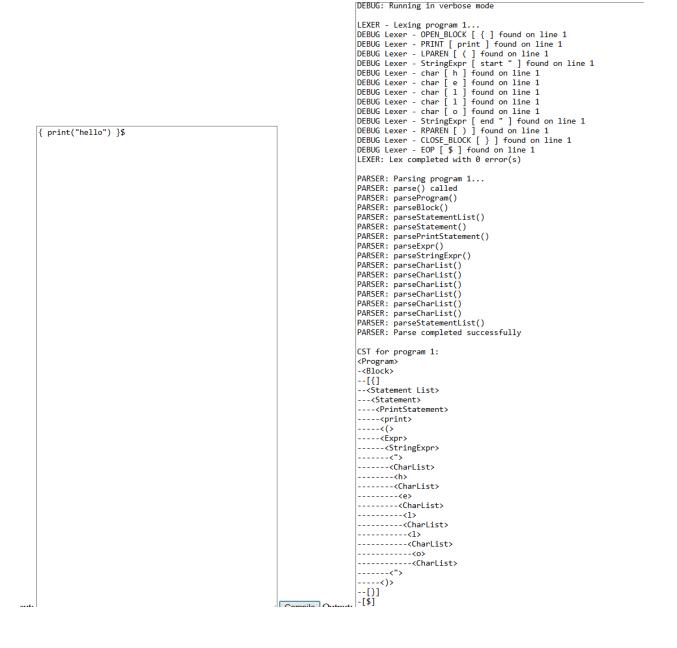
PARSER: parseStatementList()

PARSER: Parse completed successfully

CST for program 1:

<p
```

Test Case # 2:



Test Case #3:

```
{ int a }$
```

```
DEBUG: Running in verbose mode
LEXER - Lexing program 1...
DEBUG Lexer - OPEN BLOCK [ { ] found on line 1
DEBUG Lexer - ITYPE [ int ] found on line 1
DEBUG Lexer - ID [ a ] found on line 1
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 1
DEBUG Lexer - EOP [ $ ] found on line 1
LEXER: Lex completed with 0 error(s)
PARSER: Parsing program 1...
PARSER: parse() called
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseStatementList()
PARSER: Parse completed successfully
 CST for program 1:
 <Program>
 -<Block>
 --[{]
 --<Statement List>
 ---<Statement>
 ----<VarDecl>
 ----<int>
 ----<a>
 --[}]
-[$]
```

Test Case # 4:

```
{ print("Hello") }$
```

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...

DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1

DEBUG Lexer - PRINT [ print ] found on line 1

DEBUG Lexer - LPAREN [ ( ] found on line 1

DEBUG Lexer - StringExpr [ start " ] found on line 1

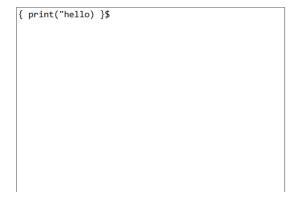
ERROR Lexer - Error: line 1 Unrecognized Token: H
Only lowercase letters a through z and spaces are allowed in strings

Error Lexer - Lex failed with 1 error(s)

PARSER: Skipped due to LEXER error(s)

CST for program 1: Skipped due to LEXER error(s).
```

Test Case # 5:



```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...

DEBUG Lexer - OPEN_BLOCK [ { } ] found on line 1

DEBUG Lexer - PRINT [ print ] found on line 1

DEBUG Lexer - LPAREN [ ( ) found on line 1

DEBUG Lexer - StringExpr [ start " ] found on line 1

ERROR Lexer - Error: Unterminated StringExpr starting on line 1. Lexing terminated due to fatal error.

Error Lexer - Lex failed with 1 error(s)

PARSER: Skipped due to LEXER error(s)

CST for program 1: Skipped due to LEXER error(s).
```

Test Case # 6:

```
{ /* This comment never ends }$
```

DEBUG: Running in verbose mode LEXER - Lexing program 1... DEBUG Lexer - OPEN_BLOCK [{ }] found on line 1 ERROR Lexer - Error: Unterminated comment starting on line 1. Lexing terminated. Error Lexer - Lex failed with 1 error(s) PARSER: Skipped due to LEXER error(s) CST for program 1: Skipped due to LEXER error(s).

Test Case #7:

```
{ int @ }$
```

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...

DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1

DEBUG Lexer - ITYPE [ int ] found on line 1

ERROR Lexer - Error: line 1 Unrecognized Token: @

Please reference grammar guide.

DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 1

DEBUG Lexer - EOP [ $ ] found on line 1

LEXER: Lex completed with 1 error(s)

PARSER: Skipped due to LEXER error(s).
```

Test Case #8:

```
{ print("test") }$
                                                                                                           DEBUG: Running in verbose mode
                                                                                                           LEXER - Lexing program 1...

DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1

DEBUG Lexer - PRINT [ print ] found on line 1

DEBUG Lexer - LPAREN [ ( ] found on line 1

DEBUG Lexer - StringExpr [ start " ] found on line
                                                                                                           DEBUG Lexer - StringExpr [ end " ] found on line 1
DEBUG Lexer - RPAREN [ ) ] found on line 1
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 1
DEBUG Lexer - EOP [ $ ] found on line 1
                                                                                                           LEXER: Lex completed with 0 error(s)
                                                                                                           PARSER: Parsing program 1...
PARSER: parse() called
                                                                                                           PARSER: parseProgram()
                                                                                                           PARSER: parseBlock()
                                                                                                           PARSER: parseStatementList()
                                                                                                           PARSER: parseStatement()
                                                                                                           PARSER: parsePrintStatement()
                                                                                                           PARSER: parseExpr()
                                                                                                           PARSER: parseStringExpr()
                                                                                                           PARSER: parseCharList()
                                                                                                           PARSER: parseStatementList()
PARSER: Parse completed successfully
                                                                                                           CST for program 1:
                                                                                                            <Program>
                                                                                                            -<Block>
                                                                                                            --[{]
--<Statement List>
                                                                                                            ---<Statement>
                                                                                                            ----<PrintStatement>
                                                                                                            ----<print>
                                                                                                            -----(>
                                                                                                            ----<Èxpr>
                                                                                                            -----<StringExpr>
                                                                                                            -----
                                                                                                            -----<CharList>
                                                                                                            -----
                                                                                                            ----<)>
                                                                                                            --[}]
-[$]
```

Test Case #9:

Test Case # 10:

```
PARSER: Parsing program 1...
PARSER: parse() called
PARSER: parse()
```

DEBUG: Running in verbose mode

LEXER - Lexing program 1...

DEBUG Lexer - OPEN_BLOCK [{] found on line 1

DEBUG Lexer - CLOSE_BLOCK [}] found on line 1

DEBUG Lexer - EOP [\$] found on line 1

LEXER: Lex completed with 0 error(s)

ıt:

```
----<PrintStatement>
-----<print>
-----(>
-----<Expr>
-----<StringExpr>
                                                                                                                                                                                    -----<">
                                                                                                                                                                                    ------(h)
-----(CharList)
-----(e)
-----(CharList)
                                                                                                                                                                                       ------<1>
------(CharList>
    {}$
{ print("hello") }$
{ int @ }$
{ print("abc) }$
{ /* comment */ int a }$
                                                                                                                                                                               ----<)>
|--[}]
-[$]
                                                                                                                                                                                LEXER - Lexing program 3...

DEBUG Lexer - OPEN_BLOCK [ { } found on line 3

DEBUG Lexer - TTYPE [ int ] found on line 3

ERROR Lexer - ITYPE [ int ] found on line 3

ERROR Lexer - From: line 3 Unrecognized Token: @ Please reference grammar guide.

DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 3

DEBUG Lexer - EOP [ $ ] found on line 3

LEXER: Lex completed with 1 error(s)
                                                                                                                                                                                PARSER: Skipped due to LEXER error(s)
CST for program 3: Skipped due to LEXER error(s).
                                                                                                                                                                               LEXER - Loxing program 4... | found on line 4

DEBUG Lexer - PRINT [prist] start "] found on line 4

DEBUG Lexer - PRINT [prist] found on line 4

DEBUG Lexer - LPAREN [ ( ) found on line 4

DEBUG Lexer - StringExpr | start "] found on line 4

DEBUG Lexer - StringExpr | start "] found on line 4

DEBUG Lexer - char [ a ] found on line 4

DEBUG Lexer - char [ a ] found on line 4

DEBUG Lexer - char [ b ] found on line 4

DEBUG Lexer - char [ c ] found on line 4

DEBUG Lexer - char [ c ] found on line 4

DEBUG Lexer - char [ c ] found on line 4

DEBUG Lexer - char [ c ] found on line 4

DEBUG Lexer - char [ c ] found on line 4

DEBUG Lexer - Fronc: line 4 Unrecognized Token in string: ) Only lowercase letters a through z and spaces are allowed in strings from lexer - Lex failed with 1 error(s)
                                                                                                                                                                                 PARSER: Skipped due to LEXER error(s)
CST for program 4: Skipped due to LEXER error(s).
                                                                                                                                                                                LEXER - Lexing program 5...
DEBUG Lexer - OPEN BLOCK [ { ] found on line 5
DEBUG Lexer - ITYPE [ int ] found on line 5
DEBUG Lexer - ID [ a ] found on line 5
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 5
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 5
LEXER: Lex completed with 0 error(s)
                                                                                                                                    PARSER: Parsing program 5...
PARSER: parse() called
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVatatement()
PARSER: parseStatementList()
PARSER: Parse completed successfully
CST for program 5:
 <Program>
 -<Block>
  --[{]
  --<Statement List>
  ---<Statement>
  ----<VarDecl>
  ----<int>
  ----<a>
  --[}]
  -[$]
```

Overall, all errors and warnings are recognized and described in detail with recommended solutions. Parsing doesn't happen if lexing fails and a CST isn't produced if parsing fails. Throughout testing I found bugs, so the testing was overall very useful! The test results above are the final results after making these final adjustments.