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: parseBlock()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1:
<Program>
-<Block>
--[{]
--<Statement List>
--[}]
-[$]
```

Test Case # 2:

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

Compile Output:

```
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
DEBUG Lexer - char [ h ] found on line 1
DEBUG Lexer - char [ e ] found on line 1
DEBUG Lexer - char [ l ] found on line 1
DEBUG Lexer - char [ l ] found on line 1
DEBUG Lexer - char [ o ] found on line 1
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: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1:
<Program>
-<Block>
--[{]
--<Statement List>
---<Statement>
----<PrintStatement>
-----<print>
-----<(>
-----<Expr>
------<StringExpr>
-------<">
-------<CharList>
--------<h>
--------<CharList>
---------<e>
---------<CharList>
----------<l>
----------<CharList>
-----------<l>
-----------<CharList>
------------<o>
------------<CharList>
-------<">
-----<)>
--[}]
-[$]
```

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:

```
{ 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: 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)
CST for program 1: 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
1
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>
-----<(>
-----<Expr>
------<StringExpr>
-------<">
-------<CharList>
-------<">
-----<)>
--[}]
-[$]
```

Test Case # 9:

```
{ print("hello") }$abc
```

Compile Output:

```
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: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1:
<Program>
-<Block>
--[{]
--<Statement List>
---<Statement>
----<PrintStatement>
-----<print>
-----<(>
-----<Expr>
------<StringExpr>
-------<">
-------<CharList>
--------<h>
--------<CharList>
---------<e>
---------<CharList>
----------<l>
----------<CharList>
-----------<l>
-----------<CharList>
------------<o>
------------<CharList>
-------<">
-----<)>
--[}]
-[$]

LEXER - Lexing program 2...
DEBUG Lexer - ID [ a ] found on line 1
DEBUG Lexer - ID [ b ] found on line 1
DEBUG Lexer - ID [ c ] found on line 1
ERROR Lexer - Error: last line of program does not end with "$".
Error Lexer - Lex failed with 1 error(s)

PARSER: Skipped due to LEXER error(s)
CST for program 2: Skipped due to LEXER error(s).
```

Test Case # 10:

```
{}$
{ print("hello") }$
{ int @ }$
{ print("abc) }$
{ /* comment */ int a }$
```

Compile

Output:

```
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: parseBlock()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1:
<Program>
-<Block>
--[{]
--<Statement List>
--[}]
-[$]

LEXER - Lexing program 2...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 2
DEBUG Lexer - PRINT [ print ] found on line 2
DEBUG Lexer - LPAREN [ ( ] found on line 2
DEBUG Lexer - StringExpr [ start " ] found on line 2
DEBUG Lexer - char [ h ] found on line 2
DEBUG Lexer - char [ e ] found on line 2
DEBUG Lexer - char [ l ] found on line 2
DEBUG Lexer - char [ l ] found on line 2
DEBUG Lexer - char [ o ] found on line 2
DEBUG Lexer - StringExpr [ end " ] found on line 2
DEBUG Lexer - RPAREN [ ) ] found on line 2
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 2
DEBUG Lexer - EOP [ $ ] found on line 2
LEXER: Lex completed with 0 error(s)

PARSER: Parsing program 2...
PARSER: parse() called
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 2:
<Program>
-<Block>
--[{]
--<Statement List>
---<Statement>
```

```
{}$
{ print("hello") }$
{ int @ }$
{ print("abc) }$
{ /* comment */ int a }$
```

Compile Output:

```
----<PrintStatement>
-----<print>
-----<(>
-----<Expr>
------<StringExpr>
-------<">
-------<CharList>
--------<h>
--------<CharList>
---------<e>
---------<CharList>
----------<l>
----------<CharList>
----------<l>
----------<CharList>
----------<o>
----------<CharList>
-------<">
-----<)>
--[}]
-[$]

LEXER - Lexing program 3...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 3
DEBUG Lexer - ITYPE [ int ] found on line 3
ERROR Lexer - Error: 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 - Lexing program 4...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 4
DEBUG Lexer - PRINT [ print ] found on line 4
DEBUG Lexer - LPAREN [ ( ] found on line 4
DEBUG Lexer - StringExpr [ start " ] 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
ERROR Lexer - Error: line 4 Unrecognized Token in string: ) 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 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 - EOP [ $ ] 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: parseVarDecl()
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.