1.

put:

```
{
  int a
  a = 5
  print(a)
}
$
```

Compile

Output:

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1
DEBUG Lexer - ITYPE [ int ] found on line 2
DEBUG Lexer - ID [ a ] found on line 2
DEBUG Lexer - ID [ a ] found on line 3
DEBUG Lexer - ASSIGN_OP [ = ] found on line 3
DEBUG Lexer - DIGIT [ 5 ] found on line 3
DEBUG Lexer - PRINT [ print ] found on line 4
DEBUG Lexer - LPAREN [ ( ] found on line 4
DEBUG Lexer - ID [ a ] found on line 4
DEBUG Lexer - RPAREN [ ) ] found on line 4
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 5
DEBUG Lexer - EOP [ $ ] found on line 6
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: parseStatement()
PARSER: parseAssignmentStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1:
<Program>
-<Block>
--[{]
--<Statement List>
---<Statement>
----<VarDecl>
-----<int>
-----<a>
---<Statement>
----<AssignmentStatement>
-----<a>
-----<=>
-----<Expr>
------<IntExpr>
-------<5>
---<Statement>
----<PrintStatement>
```

```
-----<print>
-----<(>
-----<Expr>
------<a>
-----<)>
--[}]
-[$]

AST for program 1:
< BLOCK >
-< Statement >
--< Variable Declaration >
---[ int ]
---[ a ]
--< Statement >
---< Assignment Statement >
----[ a ]
----[ 5 ]
--< Statement >
---< Print Statement >
----[ a ]

Program 1 Semantic Analysis produced
0 error(s) and 0 warning(s)

Program 1 Symbol Table
---
Name Type    Scope Line
---
a    int    0    2
```

2.

```
{
  b = 5
  print(b)
}
$
```

::

Compile

Output:

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1
DEBUG Lexer - ID [ b ] found on line 2
DEBUG Lexer - ASSIGN_OP [ = ] found on line 2
DEBUG Lexer - DIGIT [ 5 ] found on line 2
DEBUG Lexer - PRINT [ print ] found on line 3
DEBUG Lexer - LPAREN [ ( ] found on line 3
DEBUG Lexer - ID [ b ] found on line 3
DEBUG Lexer - RPAREN [ ) ] found on line 3
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 4
DEBUG Lexer - EOP [ $ ] found on line 5
LEXER: Lex completed with 0 error(s)

PARSER: Parsing program 1...
PARSER: parse() called
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignmentStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1:
<Program>
-<Block>
--[{]
--<Statement List>
---<Statement>
----<AssignmentStatement>
-----<b>
-----<=>
-----<Expr>
------<IntExpr>
-------<5>
---<Statement>
----<PrintStatement>
-----<print>
-----<(>
-----<Expr>
------<b>
-----<)>
--[}]
-[$]
```

```
AST for program 1:
< BLOCK >
-< Statement >
--< Assignment Statement >
---[ b ]
---[ 5 ]
--< Statement >
---< Print Statement >
----[ b ]

Program 1 Semantic Analysis produced
1 error(s) and 0 warning(s)
ERROR: Semantic Error: Variable 'b' used before
declaration.

Program 1 Symbol Table not produced due to
error(s) detected by semantic analysis
```

3.

```
{
  int a
  int a
}
$
```

t:

Compile

Output:

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1
DEBUG Lexer - ITYPE [ int ] found on line 2
DEBUG Lexer - ID [ a ] found on line 2
DEBUG Lexer - ITYPE [ int ] found on line 3
DEBUG Lexer - ID [ a ] found on line 3
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 4
DEBUG Lexer - EOP [ $ ] found on line 5
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: parseStatement()
PARSER: parseVarDecl()
PARSER: parseStatementList()
PARSER: Parse completed successfully

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

AST for program 1:
< BLOCK >
-< Statement >
--< Variable Declaration >
---[ int ]
---[ a ]
--< Statement >
---< Variable Declaration >
----[ int ]
----[ a ]

Program 1 Semantic Analysis produced
1 error(s) and 2 warning(s)
```

```
Program 1 Semantic Analysis produced
1 error(s) and 2 warning(s)
ERROR: Redeclaration error: 'a' already declared
in scope 0 at line 3, column 7.
WARNING: Warning: Variable 'a' declared at line
2, column 7 but never used.
WARNING: Warning: Variable 'a' declared at line
2, column 7 but never assigned a value.

Program 1 Symbol Table not produced due to
error(s) detected by semantic analysis
```

4.

```
{
  int x
}
$
```

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1
DEBUG Lexer - ITYPE [ int ] found on line 2
DEBUG Lexer - ID [ x ] found on line 2
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 3
DEBUG Lexer - EOP [ $ ] found on line 4
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>
-----<x>
--[}]
-[$]

AST for program 1:
< BLOCK >
-< Statement >
--< Variable Declaration >
---[ int ]
---[ x ]

Program 1 Semantic Analysis produced
0 error(s) and 2 warning(s)
WARNING: Warning: Variable 'x' declared at line 2,
column 7 but never used.
WARNING: Warning: Variable 'x' declared at line 2,
column 7 but never assigned a value.

Program 1 Symbol Table
---
Name Type    Scope Line
---
x    int    0    2
```

5.

```
{
  int x
  {
    int y
    y = 7
    print(y)
  }
  print(x)
}
$
```

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1
DEBUG Lexer - ITYPE [ int ] found on line 2
DEBUG Lexer - ID [ x ] found on line 2
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 3
DEBUG Lexer - ITYPE [ int ] found on line 4
DEBUG Lexer - ID [ y ] found on line 4
DEBUG Lexer - ID [ y ] found on line 5
DEBUG Lexer - ASSIGN_OP [ = ] found on line 5
DEBUG Lexer - DIGIT [ 7 ] found on line 5
DEBUG Lexer - PRINT [ print ] found on line 6
DEBUG Lexer - LPAREN [ ( ] found on line 6
DEBUG Lexer - ID [ y ] found on line 6
DEBUG Lexer - RPAREN [ ) ] found on line 6
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 7
DEBUG Lexer - PRINT [ print ] found on line 8
DEBUG Lexer - LPAREN [ ( ] found on line 8
DEBUG Lexer - ID [ x ] found on line 8
DEBUG Lexer - RPAREN [ ) ] found on line 8
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 9
DEBUG Lexer - EOP [ $ ] found on line 10
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: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignmentStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
```

```
PARSER: Parse completed successfully

CST for program 1:
<Program>
-<Block>
--[{]
--<Statement List>
---<Statement>
----<VarDecl>
-----<int>
-----<x>
---<Statement>
----<Block>
-----[{]
-----<Statement List>
------<Statement>
-------<VarDecl>
--------<int>
--------<y>
------<Statement>
-------<AssignmentStatement>
--------<y>
--------<=>
--------<Expr>
---------<IntExpr>
----------<7>
------<Statement>
-------<PrintStatement>
--------<print>
--------<(>
--------<Expr>
---------<y>
--------<)>
-----[}]
---<Statement>
----<PrintStatement>
-----<print>
-----<(>
-----<Expr>
------<x>
-----<)>
--[}]
-[$]

AST for program 1:
< BLOCK >
-< Statement >
--< Variable Declaration >
---[ int ]
---[ x ]
--< Statement >
---< BLOCK >
----< Statement >
-----< Variable Declaration >
------[ int ]
------[ y ]
-----< Statement >
------< Assignment Statement >
-------[ y ]
-------[ 7 ]
-----< Statement >
------< Print Statement >
-------[ y ]
--< Statement >
---< Print Statement >
----[ x ]

Program 1 Semantic Analysis produced
0 error(s) and 1 warning(s)
WARNING: Warning: Variable 'x' declared at line
2, column 7 but never assigned a value.

Program 1 Symbol Table
---
Name Type    Scope Line
---
x    int     0     2
y    int     1     4
```

6.

```
{
  int a
  a = "hello"
}
$
```

Compile Output:

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1
DEBUG Lexer - ITYPE [ int ] found on line 2
DEBUG Lexer - ID [ a ] found on line 2
DEBUG Lexer - ID [ a ] found on line 3
DEBUG Lexer - ASSIGN_OP [ = ] found on line 3
DEBUG Lexer - StringExpr [ start " ] found on
line 3
DEBUG Lexer - char [ h ] found on line 3
DEBUG Lexer - char [ e ] found on line 3
DEBUG Lexer - char [ l ] found on line 3
DEBUG Lexer - char [ l ] found on line 3
DEBUG Lexer - char [ o ] found on line 3
DEBUG Lexer - StringExpr [ end " ] found on line
3
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 4
DEBUG Lexer - EOP [ $ ] found on line 5
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: parseStatement()
PARSER: parseAssignmentStatement()
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>
----<VarDecl>
-----<int>
-----<a>
---<Statement>
----<AssignmentStatement>
-----<a>
```

```
-----<=>
-----<Expr>
------<StringExpr>
-------<">
-------<CharList>
--------<h>
--------<CharList>
---------<e>
---------<CharList>
----------<l>
----------<CharList>
-----------<l>
-----------<CharList>
------------<o>
------------<CharList>
-------------<Îµ>
-------<">
--[}]
-[$]

AST for program 1:
< BLOCK >
-< Statement >
--< Variable Declaration >
---[ int ]
---[ a ]
--< Statement >
---< Assignment Statement >
----[ a ]
----[ hello ]

Program 1 Semantic Analysis produced
1 error(s) and 0 warning(s)
ERROR: Semantic Error: Type mismatch in
assignment to 'a', found string.

Program 1 Symbol Table not produced due to
error(s) detected by semantic analysis
```

7.

```
{
  if (5) {
    print("bad")
  }
}

$
```

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1
DEBUG Lexer - IFSTATEMENT [ if ] found on line 2
DEBUG Lexer - LPAREN [ ( ] found on line 2
DEBUG Lexer - DIGIT [ 5 ] found on line 2
DEBUG Lexer - RPAREN [ ) ] found on line 2
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 2
DEBUG Lexer - PRINT [ print ] found on line 3
DEBUG Lexer - LPAREN [ ( ] found on line 3
DEBUG Lexer - StringExpr [ start " ] found on line
3
DEBUG Lexer - char [ b ] found on line 3
DEBUG Lexer - char [ a ] found on line 3
DEBUG Lexer - char [ d ] found on line 3
DEBUG Lexer - StringExpr [ end " ] found on line 3
DEBUG Lexer - RPAREN [ ) ] found on line 3
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 4
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 5
DEBUG Lexer - EOP [ $ ] found on line 7
LEXER: Lex completed with 0 error(s)

PARSER: Parsing program 1...
PARSER: parse() called
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseIfStatement()
PARSER: parseBooleanExpr()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseBoolOp()

PARSER: Parse failed with 1 error

PARSER ERROR: PARSER ERROR: Expected boolean
operator at line 2
CST for program 1: Skipped due to PARSER error(s).
```

8.

```
{
  boolean d
  d = false
  while (d != true) {
    print("looping")
  }
}
$
```

Compile Output:

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1
DEBUG Lexer - ITYPE [ boolean ] found on line 2
DEBUG Lexer - ID [ d ] found on line 2
DEBUG Lexer - ID [ d ] found on line 3
DEBUG Lexer - ASSIGN_OP [ = ] found on line 3
DEBUG Lexer - BOOLVALF [ false ] found on line 3
DEBUG Lexer - WHILE [ while ] found on line 4
DEBUG Lexer - LPAREN [ ( ] found on line 4
DEBUG Lexer - ID [ d ] found on line 4
DEBUG Lexer - BOOL_INEQUAL [ != ] found on line 4
DEBUG Lexer - BOOLVALT [ true ] found on line 4
DEBUG Lexer - RPAREN [ ) ] found on line 4
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 4
DEBUG Lexer - PRINT [ print ] found on line 5
DEBUG Lexer - LPAREN [ ( ] found on line 5
DEBUG Lexer - StringExpr [ start " ] found on
line 5
DEBUG Lexer - char [ l ] found on line 5
DEBUG Lexer - char [ o ] found on line 5
DEBUG Lexer - char [ o ] found on line 5
DEBUG Lexer - char [ p ] found on line 5
DEBUG Lexer - char [ i ] found on line 5
DEBUG Lexer - char [ n ] found on line 5
DEBUG Lexer - char [ g ] found on line 5
DEBUG Lexer - StringExpr [ end " ] found on line
5
DEBUG Lexer - RPAREN [ ) ] found on line 5
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 6
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 7
DEBUG Lexer - EOP [ $ ] found on line 8
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: parseStatement()
PARSER: parseAssignmentStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseWhileStatement()
PARSER: parseBooleanExpr()
PARSER: parseExpr()
PARSER: parseBoolOp()
PARSER: parseExpr()
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: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1:
<Program>
-<Block>
--[{]
--<Statement List>
---<Statement>
----<VarDecl>
-----<boolean>
-----<d>
---<Statement>
----<AssignmentStatement>
-----<d>
-----<=>
-----<Expr>
------<false>
---<Statement>
----<WhileStatement>
-----<while>
-----<BooleanExpr>
------<(>
------<Expr>
-------<d>
------<BoolOp>
-------<!=>
------<Expr>
-------<true>
------<)>
-----<Block>
------[{]
------<Statement List>
-------<Statement>
--------<PrintStatement>
---------<print>
---------<(>
---------<Expr>
----------<StringExpr>
----------<">
```

```
----------<CharList>
------------<l>
------------<CharList>
-------------<o>
-------------<CharList>
--------------<o>
--------------<CharList>
---------------<p>
---------------<CharList>
----------------<i>
----------------<CharList>
-----------------<n>
-----------------<CharList>
------------------<g>
------------------<CharList>
-------------------<Îµ>
-----------<">
---------<)>
------[}]
--[}]
-[$]

AST for program 1:
< BLOCK >
-< Statement >
--< Variable Declaration >
---[ boolean ]
---[ d ]
--< Statement >
---< Assignment Statement >
----[ d ]
----[ false ]
--< Statement >
---< While Statement >
----[ while ]
----< BooleanExpr >
-----< Expr >
------[ d ]
-----< BoolOp >
------[ != ]
-----< Expr >
------[ true ]
----< BLOCK >
-----< Statement >
------< Print Statement >
-------[ looping ]

Program 1 Semantic Analysis produced
0 error(s) and 0 warning(s)

Program 1 Symbol Table
---
Name Type    Scope Line
---
d    bool    0     2
```

9.

```
{
  while (5) {
    print("oops")
  }
}
$
```

```
DEBUG: Running in verbose mode

LEXER - Lexing program 1...
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 1
DEBUG Lexer - WHILE [ while ] found on line 2
DEBUG Lexer - LPAREN [ ( ] found on line 2
DEBUG Lexer - DIGIT [ 5 ] found on line 2
DEBUG Lexer - RPAREN [ ) ] found on line 2
DEBUG Lexer - OPEN_BLOCK [ { ] found on line 2
DEBUG Lexer - PRINT [ print ] found on line 3
DEBUG Lexer - LPAREN [ ( ] found on line 3
DEBUG Lexer - StringExpr [ start " ] found on line
3
DEBUG Lexer - char [ o ] found on line 3
DEBUG Lexer - char [ o ] found on line 3
DEBUG Lexer - char [ p ] found on line 3
DEBUG Lexer - char [ s ] found on line 3
DEBUG Lexer - StringExpr [ end " ] found on line 3
DEBUG Lexer - RPAREN [ ) ] found on line 3
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 4
DEBUG Lexer - CLOSE_BLOCK [ } ] found on line 5
DEBUG Lexer - EOP [ $ ] found on line 6
LEXER: Lex completed with 0 error(s)

PARSER: Parsing program 1...
PARSER: parse() called
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseWhileStatement()
PARSER: parseBooleanExpr()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseBoolOp()

PARSER: Parse failed with 1 error

PARSER ERROR: PARSER ERROR: Expected boolean
operator at line 2
CST for program 1: Skipped due to PARSER error(s).
```

This testing process revealed many bugs too me that I had to go back and change. These test cases now completely and correctly show code that should and shouldn’t run!