Nicholas Carmello
Design of Compilers
Monday, March 14th

Test Case:

{boolean b
b =3}$

OutPut:

<Program>
-<Block>
--[{]
--<Statement List>
---<statement>
----<VarDecl>
-----[boolean]
-----<ID>
------[b]
---<Statement List>
----<statement>
-----<Assignment Statement>
------<ID>
-------[b]
------[=]
------<Expression>
-------<Int Expr>
--------<Digit>
---------[3]
--[}]
-[$]

Test Case:

```
{}$
 {{{{{}}}}}$
 {{{{{}}} /* comments are ignored */ }}}$
{ /* comments are still ignored */ int @}$
```

Output:

```
<Program>
-<Block>
--[{]
--[Statement List]
--[}]
-[$]

<Program>
-<Block>
--[{]
--<Statement List>
---<statement>
----<Block>
-----[{]
-----<Statement List>
------<statement>
-------<Block>
--------[{]
--------<Statement List>
---------<statement>
----------<Block>
-----------[{]
-----------<Statement List>
------------<statement>
-------------<Block>
--------------[{]
--------------<Statement List>
---------------<statement>
----------------<Block>
-----------------[{]
-----------------[Statement List]
-----------------[}]
```

----------[}]
----------[}]
--------[}]
-----[}]
--[}]
-[$]

<Program>
-<Block>
--[{]
--<Statement List>
---<statement>
----<Block>
-----[{]
-----<Statement List>
------<statement>
-------<Block>
--------[{]
--------<Statement List>
---------<statement>
----------<Block>
----------[{]
----------<Statement List>
----------<statement>
----------<Block>
----------[{]
----------<Statement List>
----------<statement>
----------<Block>
----------[{]
----------[Statement List]
----------[}]
----------[}]
----------[}]
--------[}]
-----[}]
--[}]
-[$]

It did not parse the fourth test case

Test Case:
/* Test case for invalid StatementList */
{
4 + 2
}$

Output:
INFO LEXER - Lexing program 1
DEBUG LEXER - Left Curly [{] found at line: 2, position: 1
DEBUG LEXER - Type Num [4] found at line: 3, position: 1
DEBUG LEXER - Addition Op [+] found at line: 3, position: 3
DEBUG LEXER - Type Num [2] found at line: 3, position: 5
DEBUG LEXER - Right Curly [}] found at line: 4, position: 1
DEBUG LEXER - EOP [$] found at line: 4, position: 2
INFO LEXER - Lex Passed with 0 errors!!!

DEBUG PARSER - SUCCESS - Expected: Left Curly, Received: {
DEBUG PARSER - ERROR - Expected: Right Curly, Received: 4
INFO PARSER - Parser failed. Not Printing CST.

Test Case:
/* hello world

Output:

INFO LEXER - Lexing program 1
INFO LEXER - No $ at the end of the program. Adding One.
ERROR LEXER - The Comment was never terminated or '$' was in the comment at line 1, position: 15
ERROR LEXER - Lex failed with 1 error(s)
Not going to parse

Test Case:
/* Extra Right Brace */
{{{{{{}}} /* comments are ignored */ }}}}$

Output:

```
INFO LEXER - Lexing program 1
DEBUG LEXER - Left Curly [ { ] found at line: 2, position: 1
DEBUG LEXER - Left Curly [ { ] found at line: 2, position: 2
DEBUG LEXER - Left Curly [ { ] found at line: 2, position: 3
DEBUG LEXER - Left Curly [ { ] found at line: 2, position: 4
DEBUG LEXER - Left Curly [ { ] found at line: 2, position: 5
DEBUG LEXER - Left Curly [ { ] found at line: 2, position: 6
DEBUG LEXER - Right Curly [ } ] found at line: 2, position: 7
DEBUG LEXER - Right Curly [ } ] found at line: 2, position: 8
DEBUG LEXER - Right Curly [ } ] found at line: 2, position: 9
DEBUG LEXER - Right Curly [ } ] found at line: 2, position: 38
DEBUG LEXER - Right Curly [ } ] found at line: 2, position: 39
DEBUG LEXER - Right Curly [ } ] found at line: 2, position: 40
DEBUG LEXER - Right Curly [ } ] found at line: 2, position: 41
DEBUG LEXER - EOP [ $ ] found at line: 2, position: 42
INFO LEXER - Lex Passed with 0 errors!!!

DEBUG PARSER - SUCCESS - Expected: Left Curly, Recieved: {
DEBUG PARSER - SUCCESS - Expected: Left Curly, Recieved: {
DEBUG PARSER - SUCCESS - Expected: Left Curly, Recieved: {
DEBUG PARSER - SUCCESS - Expected: Left Curly, Recieved: {
DEBUG PARSER - SUCCESS - Expected: Left Curly, Recieved: {
DEBUG PARSER - SUCCESS - Expected: Left Curly, Recieved: {
DEBUG PARSER - SUCCESS - Expected: Right Curly, Recieved: }
DEBUG PARSER - SUCCESS - Expected: Right Curly, Recieved: }
DEBUG PARSER - SUCCESS - Expected: Right Curly, Recieved: }
DEBUG PARSER - SUCCESS - Expected: Right Curly, Recieved: }
DEBUG PARSER - SUCCESS - Expected: Right Curly, Recieved: }
DEBUG PARSER - SUCCESS - Expected: Right Curly, Recieved: }
DEBUG PARSER - ERROR - Expected: EOP, Recieved: }
INFO PARSER - Parser failed. Not Printing CST.
```

Test Case:

```
/* Test case for IfStatement. Prints numsidsstringsbooleans */
{
   int a
   a = 1
   if(1 == 1){
      print(n)
   }
   if(3 == 3){
```

```
        print(3)
    }
    if(3 == 3){
        print(s)
    }
    if(true == true){
        print(s)
    }
} $
```

Output:

```
<Program>
-<Block>
--[{]
--<Statement List>
---<statement>
----<VarDecl>
-----<Type Int>
------[int]
-----<ID>
------[a]
---<Statement List>
----<statement>
-----<Assignment Statement>
------<ID>
-------[a]
------[=]
------<Expression>
-------<Int Expr>
--------<Digit>
---------[1]
----<Statement List>
-----<statement>
------<If Statement>
-------[if]
-------<Bool Expr>
--------[(]
--------<Expression>
---------<Int Expr>
----------<Digit>
-----------[1]
--------<Bool Op>
---------[==]
```

--------<Expression>
---------<Int Expr>
----------<Digit>
-----------[1]
--------[)]
-------<Block>
--------[{]
--------<Statement List>
---------<statement>
----------<Print>
-----------[print]
-----------[(]
-----------<Expression>
------------<ID>
-------------[n]
-----------[)]
--------[}]
-----<Statement List>
------<statement>
-------<If Statement>
--------[if]
--------<Bool Expr>
---------[(]
---------<Expression>
----------<Int Expr>
-----------<Digit>
------------[3]
---------<Bool Op>
----------[==]
---------<Expression>
----------<Int Expr>
-----------<Digit>
------------[3]
---------[)]
--------<Block>
---------[{]
---------<Statement List>
----------<statement>
-----------<Print>
------------[print]
-----------[(]
------------<Expression>
-------------<Int Expr>
--------------<Digit>

----------[3]
----------[)]
---------[}]
------<Statement List>
-------<statement>
--------<If Statement>
---------[if]
---------<Bool Expr>
----------[(]
----------<Expression>
-----------<Int Expr>
------------<Digit>
------------[3]
----------<Bool Op>
-----------[==]
----------<Expression>
-----------<Int Expr>
------------<Digit>
------------[3]
----------[)]
---------<Block>
----------[{]
----------<Statement List>
-----------<statement>
------------<Print>
------------[print]
------------[(]
------------<Expression>
--------------<ID>
---------------[s]
------------[)]
----------[}]
-------<Statement List>
--------<statement>
---------<If Statement>
----------[if]
----------<Bool Expr>
-----------[(]
-----------<Expression>
------------<Bool Expr>
------------[true]
-----------<Bool Op>
------------[==]
-----------<Expression>

----------<Bool Expr>
----------[true]
----------[)]
----------<Block>

----------[{]
----------<Statement List>
----------<statement>
----------<Print>
----------[print]

----------[(]
----------<Expression>
----------<ID>
----------[s]
----------[)]
----------[}]
--[}]
-[$]

Test Case:

{intaintba=0b=0while(a!=3){print(a)while(b!=3){print(b)b=1+bif(b==2){print("there isno spoon")}}b=0a=1+a}}$

Output:

INFO LEXER - Lexing program 1
DEBUG LEXER - Left Curly [{] found at line: 1, position: 1
DEBUG LEXER - Type Int [int] found at line: 1, position: 2
DEBUG LEXER - ID [a] found at line: 1, position: 5
DEBUG LEXER - Type Int [int] found at line: 1, position: 6
DEBUG LEXER - ID [b] found at line: 1, position: 9
DEBUG LEXER - ID [a] found at line: 1, position: 10
DEBUG LEXER - Assignment Op [=] found at line: 1, position: 11
DEBUG LEXER - Type Num [0] found at line: 1, position: 12
DEBUG LEXER - ID [b] found at line: 1, position: 13
DEBUG LEXER - Assignment Op [=] found at line: 1, position: 14
DEBUG LEXER - Type Num [0] found at line: 1, position: 15
DEBUG LEXER - While statement [while] found at line: 1, position: 16
DEBUG LEXER - Left Paren [( ] found at line: 1, position: 21

DEBUG LEXER - ID [a] found at line: 1, position: 22
DEBUG LEXER - Not Equals [!=] found at line: 1, character: 23
DEBUG LEXER - Type Num [3] found at line: 1, position: 25
DEBUG LEXER - Right Paren [ )] found at line: 1, position: 26
DEBUG LEXER - Left Curly [{] found at line: 1, position: 27
DEBUG LEXER - Print Statement [print] found at line: 1, position: 28
DEBUG LEXER - Left Paren [( ] found at line: 1, position: 33
DEBUG LEXER - ID [a] found at line: 1, position: 34
DEBUG LEXER - Right Paren [ )] found at line: 1, position: 35
DEBUG LEXER - While statement [while] found at line: 1, position: 36
DEBUG LEXER - Left Paren [( ] found at line: 1, position: 41
DEBUG LEXER - ID [b] found at line: 1, position: 42
DEBUG LEXER - Not Equals [!=] found at line: 1, character: 43
DEBUG LEXER - Type Num [3] found at line: 1, position: 45
DEBUG LEXER - Right Paren [ )] found at line: 1, position: 46
DEBUG LEXER - Left Curly [{] found at line: 1, position: 47
DEBUG LEXER - Print Statement [print] found at line: 1, position: 48
DEBUG LEXER - Left Paren [( ] found at line: 1, position: 53
DEBUG LEXER - ID [b] found at line: 1, position: 54
DEBUG LEXER - Right Paren [ )] found at line: 1, position: 55
DEBUG LEXER - ID [b] found at line: 1, position: 56
DEBUG LEXER - Assignment Op [=] found at line: 1, position: 57
DEBUG LEXER - Type Num [1] found at line: 1, position: 58
DEBUG LEXER - Addition Op [+] found at line: 1, position: 59
DEBUG LEXER - ID [b] found at line: 1, position: 60
DEBUG LEXER - If Statement [if] found at line: 1, position: 61
DEBUG LEXER - Left Paren [( ] found at line: 1, position: 63
DEBUG LEXER - ID [b] found at line: 1, position: 64
DEBUG LEXER - Equals To [==] found at line: 1, character: 65
DEBUG LEXER - Type Num [2] found at line: 1, position: 67
DEBUG LEXER - Right Paren [ )] found at line: 1, position: 68
DEBUG LEXER - Left Curly [{] found at line: 1, position: 69
DEBUG LEXER - Print Statement [print] found at line: 1, position: 70
DEBUG LEXER - Left Paren [( ] found at line: 1, position: 75
DEBUG LEXER - String [there isno spoon] found at line: 1, position: 77
DEBUG LEXER - Right Paren [ )] found at line: 1, position: 94
DEBUG LEXER - Right Curly [}] found at line: 1, position: 95
DEBUG LEXER - Right Curly [}] found at line: 1, position: 96
DEBUG LEXER - ID [b] found at line: 1, position: 97
DEBUG LEXER - Assignment Op [=] found at line: 1, position: 98
DEBUG LEXER - Type Num [0] found at line: 1, position: 99
DEBUG LEXER - ID [a] found at line: 1, position: 100
DEBUG LEXER - Assignment Op [=] found at line: 1, position: 101
DEBUG LEXER - Type Num [1] found at line: 1, position: 102

DEBUG LEXER - Addition Op [+] found at line: 1, position: 103
DEBUG LEXER - ID [a] found at line: 1, position: 104
DEBUG LEXER - Right Curly [}] found at line: 1, position: 105
DEBUG LEXER - Right Curly [}] found at line: 1, position: 106
DEBUG LEXER - EOP [$] found at line: 1, position: 107
INFO LEXER - Lex Passed with 0 errors!!!

DEBUG PARSER - SUCCESS - Expected: Left Curly, Recieved: {
DEBUG PARSER - SUCCESS - Expected: Type Int, Recieved: int
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: a
DEBUG PARSER - SUCCESS - Expected: Type Int, Recieved: int
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: b
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: a
DEBUG PARSER - SUCCESS - Expected: Assignment Op, Recieved: =
DEBUG PARSER - SUCCESS - Expected: Type Num, Recieved: 0
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: b
DEBUG PARSER - SUCCESS - Expected: Assignment Op, Recieved: =
DEBUG PARSER - SUCCESS - Expected: Type Num, Recieved: 0
DEBUG PARSER - SUCCESS - Expected: While statement, Recieved: while
DEBUG PARSER - SUCCESS - Expected: Left Paren, Recieved: (
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: a
DEBUG PARSER - SUCCESS - Expected: Not Equals, Recieved: !=
DEBUG PARSER - SUCCESS - Expected: Type Num, Recieved: 3
DEBUG PARSER - SUCCESS - Expected: Right Paren, Recieved:)
DEBUG PARSER - SUCCESS - Expected: Left Curly, Recieved: {
DEBUG PARSER - SUCCESS - Expected: Print Statement, Recieved: print
DEBUG PARSER - SUCCESS - Expected: Left Paren, Recieved: (
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: a
DEBUG PARSER - SUCCESS - Expected: Right Paren, Recieved:)
DEBUG PARSER - SUCCESS - Expected: While statement, Recieved: while
DEBUG PARSER - SUCCESS - Expected: Left Paren, Recieved: (
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: b
DEBUG PARSER - SUCCESS - Expected: Not Equals, Recieved: !=
DEBUG PARSER - SUCCESS - Expected: Type Num, Recieved: 3
DEBUG PARSER - SUCCESS - Expected: Right Paren, Recieved:)
DEBUG PARSER - SUCCESS - Expected: Left Curly, Recieved: {
DEBUG PARSER - SUCCESS - Expected: Print Statement, Recieved: print
DEBUG PARSER - SUCCESS - Expected: Left Paren, Recieved: (
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: b
DEBUG PARSER - SUCCESS - Expected: Right Paren, Recieved:)
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: b
DEBUG PARSER - SUCCESS - Expected: Assignment Op, Recieved: =
DEBUG PARSER - SUCCESS - Expected: Type Num, Recieved: 1
DEBUG PARSER - SUCCESS - Expected: Addition Op, Recieved: +

DEBUG PARSER - SUCCESS - Expected: ID, Recieved: b
DEBUG PARSER - SUCCESS - Expected: If Statement, Recieved: if
DEBUG PARSER - SUCCESS - Expected: Left Paren, Recieved: (
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: b
DEBUG PARSER - SUCCESS - Expected: Equals To, Recieved: ==
DEBUG PARSER - SUCCESS - Expected: Type Num, Recieved: 2
DEBUG PARSER - SUCCESS - Expected: Right Paren, Recieved:)
DEBUG PARSER - SUCCESS - Expected: Left Curly, Recieved: {
DEBUG PARSER - SUCCESS - Expected: Print Statement, Recieved: print
DEBUG PARSER - SUCCESS - Expected: Left Paren, Recieved: (
DEBUG PARSER - SUCCESS - Expected: Type String, Recieved: string
DEBUG PARSER - SUCCESS - Expected: Right Paren, Recieved:)
DEBUG PARSER - SUCaCESS - Expected: Right Curly, Recieved: }
DEBUG PARSER - SUCCESS - Expected: Right Curly, Recieved: }
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: b
DEBUG PARSER - SUCCESS - Expected: Assignment Op, Recieved: =
DEBUG PARSER - SUCCESS - Expected: Type Num, Recieved: 0
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: a
DEBUG PARSER - SUCCESS - Expected: Assignment Op, Recieved: =
DEBUG PARSER - SUCCESS - Expected: Type Num, Recieved: 1
DEBUG PARSER - SUCCESS - Expected: Addition Op, Recieved: +
DEBUG PARSER - SUCCESS - Expected: ID, Recieved: a
DEBUG PARSER - SUCCESS - Expected: Right Curly, Recieved: }
DEBUG PARSER - SUCCESS - Expected: Right Curly, Recieved: }
DEBUG PARSER - SUCCESS - Expected: EOP, Recieved: $
INFO PARSER - Parser Passed. Printing CST.

Test Case:
{int a
a = 3}$

Output:

<Program>
-<Block>
--[{]
--<Statement List>
---<statement>
----<VarDecl>
-----[boolean]
-----<ID>
------[b]
---<Statement List>
----<statement>
-----<Assignment Statement>
------<ID>
-------[b]
------[=]
------<Expression>
-------<Int Expr>
--------<Digit>
---------[3]
--[}]
-[$]

<Program>
-<Block>
--[{]
--<Statement List>
---<statement>
----<VarDecl>
-----<Type Int>
------[int]
-----<ID>
------[a]
---<Statement List>
----<statement>
-----<Assignment Statement>
------<ID>

-------[a]
------[=]
------<Expression>
-------<Int Expr>
--------<Digit>
---------[3]
--[}]
-[$]