Test Cases

Kai Wong

04/03/2018

1 Valid Test Cases

1.1 Simple 1

```
1        /* Simple test case */
2        {}$
```

Results: Valid

1.2 Simple 2

```
1        /* Test case for print statement */
2        {
3            print("i love compilers")
4        }$
```

Results: Valid

1.3 Regular

```
1        /* Test case for a 'regular' program*/
2        {
3            int a
4            a = 1
5            print(a)
6            boolean b
7            b = true
8            print(b)
9
10           {
11               int a
12               a = 2
13               print(a)
14           }
15
16           {
17               int a
18               a = 3
19               print(a)
20           }
21
22           string s
23           s = "stra"
24           print(s)
25
26           s = "strb"
27           print(s)
28
29           if (a != 5) {
```

```
30                  print("true")
31              }
32
33              if (a == 5) {
34                  print("false")
35              }
36          }$
```

Results: Valid

1.4 Multiple

```
1           /* Test case for multiple programs */
2           {
3               print("i love compilers")
4               int a
5               a = 2
6               string s
7               s = "ha"
8           }$
9
10          {
11              int b
12              b = 4
13              string s
14              s = "hey"
15          }$
```

Results: Valid

1.5 All Productions thx Tien

```
1           /* Test case for all productions - thx Tien */
2           {
3             /* Int Declaration */
4             int a
5             int b
6             string s
7             boolean z
8
9             z = true
10            s = "kai sucks"
11
12            a = 0
13            b = 0
14
15            /* While Loop */
16            while (a != 3) {
17                print(a)
18                while (b != 3) {
19                      print(b)
20                      b = 1 + b
21                      if (b == 2) {
22                        /* Print Statement */
23                          print("kai sucks"/* This will do nothing */)
24                      }
25                }
26
27                b = 0
28                a = 1 + a
29            }
30          }$
```

Results: Valid

1.6 Crazy One Liner (Lex Pass)

```
/* Test case for crazy one liner */
+${hellotruefalse!======trueprinta=3b=0print("false true")whi33leiftruefalsestring!=
     stringintbooleanaa truewhileif{hi+++==!==}}/*aaahaha*/hahahahaha/*awao*/$
```

Results: Valid (for Lex)

1.7 Crazy One Liner Pt. 2 Thx Tien

```
/*Test case for all productions - thx Tien*/{/*IntDeclaration*/
    intaintbstringsbooleanzz=trues="kai sucks"a=0b=0/*WhileLoop*/while(a!=3){print(a
    )while(b!=3){print(b)b=1+bif(b==2){/*PrintStatement*/print("kai sucks"/*
    Thiswilldonothing*/)}}b=0a=1+a}}$
```

Results: Valid

1.8 WhileStatement

```
/* Test case for WhileStatement */
{
    string s
    int a
    a = 1
    {
        s = "hey there sexy"
        int a
        a = 2
        print(a)
    }
    {
        while (a != 5) {
            a = 1 + a
            print(a)
        }
        print(3 + a)
        print(s)
    }
} $
```

1.9 IfStatement

```
/* Test case for IfStatement */
{
    int a
    a = 1
    if(1 == 1){
        print("nums")
    }
    if(a == a){
        print("ids")
    }
    if("hey" == "hey"){
        print("strings")
    }
    if(true == (a == a)){
        print("booleans")
    }
} $
```

1.10 Infinite Loop and Max Memory

```
        /* This code segment uses the max
         - allotted memory 256 bytes
         - Also this is an infinite loop. Credit: Tien */
        {
          int a
          a = 1

          if("a" == "a") {
            a = 2
            print("a now is two")
          }

          if(a != 1) {
            a = 3
            print(" a now is three")
          }

          if(a == 1) {
            a = 3
            print("this does not print")
          }

          while true {
              print(" this will always be true hahahahahahaha")
          }

          if false {
              print("this")
          }
        } $

        \end{lstListing}
    \subsection{Boolean Expressions}
        \begin{lstlisting}[frame=single]
        /* Boolean Expr Printing: This test case
         - demonstrates the compiler's ability to
         - generate code for computing the result
         - of a BooeleanExpr and printing the result
         - Result: falsefalsetruetruetruetruefalsefalsefalsetrue
         - Credit: Tien */
        {
            boolean a
            a = false
            print((a == true))
            print((true == a))
            print((a == false))
            print((false == a))
            print((a != true))
            print((true != a))
            print((a != false))
            print((false != a))
            print(a)
            if (a == false) {
                a = true
            }
            print(a)
        }$

        \end{lstListing}

    \subsection{Variable Addition}
        \begin{lstlisting}[frame=single]
        /*
        Demonstrates compiler's ability to generate code that properly handles variable
            addition
        Credit: Tien
```

```
        */
        {
            int a
            a = 1
            int b
            b = 1
            b = 1 + a
            while (2 + a != 3 + b) {
                a = 1 + a
                print("int a is ")
                print(a)
                print(" ")
            }
            print("int b is ")
            print(b)
        }$


        \end{lstListing}

    \subsection{Addition Checking and Long Addition}
        \begin{lstlisting}[frame=single]
        /* This statement shows that addition
         - checking and printing are both valid
         - options that can be performed. Credit: Tien
         - Result: 666addition checkfalse*/
        {
            int a
            while (a != 3) {
                print(1 + 2 + 3)
                a = 1 + a
            }
            if (1+1+1+1+1 == 2+3) {
                print("addition check")
            }
            if (1+5+3 != 8) {
                print(false)
            }
        } $

        \end{lstListing}

\section{Warning Test Cases}
    \subsection{Missing EOP}
        \begin{lstlisting}[frame=single]
        /* Missing EOP */
        {
            int b
            b = 4
            string s
            s = "hey"
        }
```

Results: WARNING: No EOP [$] detected at end-of-file. Adding to end-of-file...

1.11 Semantic Warnings

```
        /* has unused and undeclared variables */
        {
            int a
            int b
            a = 3
            b = 4
            {
                string a
                a = "hey"
                print(a)
```

```
                print(b)
            }
            print(b)
            string s
            {
                boolean b
                b = false
            }
            string r
            r = "hey"
            int d
            print(d)
            d = 3
        }$
```

Results:
WARNING - Variable [d] on line 22 col 10 has been used before being initialized.
WARNING - Variable [a] on line 3 col 4 has been initialized but is not used.
WARNING - Variable [s] on line 14 col 4 has been declared but is not initialized properly.
WARNING - Variable [r] on line 19 col 4 has been initialized but is not used.
WARNING - Variable [b] on line 16 col 8 has been initialized but is not used.

2 Lex Fail Programs

2.1 Alan

```
/*  Provided By
  - Compiler Tyrant
  - Alan G Labouseur
*/
{}$
{{{{{{}}}}}}$
{{{{{{}}}}}}$
{int  @}$
```

Results: ERROR: Unrecognized or Invalid Token [@] on line 8 col 5

2.2 Invalid String 1

```
/* Test case for placing $ in quotes */
{
    print("i love com$pilers")
    int a
    a = 2
    string s
    s = "ha"
    "
}$
```

Results: ERROR: Invalid character in String [$] on line 3 col 21

2.3 Invalid String 2

```
/* Test case for invalid characters in string */
{
    string s
    s = "cookies & cream"
}$
```

Results: ERROR: Invalid character in String [&] on line 4 col 17

2.4 Invalid String 3

```
1         /* Test case for placing \n in quotes */
2         {
3             "hey
4             there"
5         }$
```

Results: ERROR: Invalid character in String [\n] on line 3 col 8

2.5 Invalid String 4

```
1         /* Test case for missing ending quote */
2         int a
3         a = 4
4         string s
5         s = "hey there
```

Results: ERROR: Missing ending quote for String literal starting on line 5 col 4

2.6 Invalid Print

```
1         /* Test case for invalid print */
2         {
3             print("my name is 11")
4         }$
```

Results: ERROR: Invalid character in String [1] on line 3 col 22

2.7 Missing End Comment Brace

```
1         /* Test case for missing end comment brace */
2         {
3             print("my name is eleven")
4             /* hey i love compilers
5         }$
```

Results: ERROR: Missing ending comment brace (*/) for comment starting on line 4 col 4

3 Parse Fail Programs

3.1 Invalid StatementList

```
1         /* Test case for invalid StatementList */
2         {
3             4 + 2
4         }$
```

Results: ERROR - Expecting [TRbrace], found [TDigit] on line 3

3.2 Invalid Expr

```
1         /* Test case for invalid Expr */
2         {
3             int a
4             a = a + 2
5         }$
```

Results: ERROR - Expecting [TRbrace], found [TDigit] on line 3

3.3 Invalid VarDecl

```
/* Test case for invalid VarDecl */
{
    int 4
}$
```

Results: ERROR - Expecting [Id], found [TDigit] on line 3

3.4 Invalid Print Pt. 2

```
/* Test case for invalid Print pt. 2 */
{
    print("$)
}$
```

Results: ERROR - Expecting [Expr], found [TEop] on line 3
ERROR - Expecting [Block], found [TRparen] on line 3

3.5 Incomplete BooleanExpr

```
/* Test case for incomplete BooleanExpr */
{
    s = "strb"
    print(s)

    if (a != ) {
        print("true")
    }
}$
```

Results: ERROR - Expecting [Expr], found [TRparen] on line 6

3.6 Incomplete IntExpr

```
/* Test case for incomplete IntExpr */
{
    int a
    a = 1 +
    print(a)
}$
```

Results: ERROR - Expecting [Expr], found [TPrint] on line 5

4 Semantic Analysis Fail Programs

4.1 Undeclared Variable

```
/* Variables being used but not declared first */
{
    int a
    b = 4
}$
```

Results: ERROR: Variable [b] on line 4 col 12 has not been previously declared.

4.2 Duplicate Variable

```
/* Variables being declared again in same scope*/
{
    int a
    {
        string a
        a = "this is fine"
    }
    boolean a /* this is not fine" */
}$
```

Results: ERROR: Variable [a] on line 8 col 20 has already been declared in current scope at line 3 col 12

4.3 Type Mismatch

```
/* A variable's type is not compatible with its assignment*/
{
    string s
    s = 4 + 3
}$
```

Results: ERROR: The variable [s] declared on line 4 col 12 is of type string and does not match the assignment type of int

4.4 Incorrect Type Comparisons

```
/* Types do not match in Boolean comparison*/
{
    if(4 == false){
        print("this no good")
    }
    if(4 == "hey"){
        print("int to string")
    }
    if(false != "hey"){
        print("bool to string")
    }
    if(4 != 3){
        print("int to int")
    }
}$
```

Results: ERROR: The [Expression] on line 3 col 15 is of type int and is incompatibly compared to a type of boolean

4.5 Incorrect Integer Expression

```
/* A digit is added to something other than a digit */
{
    int a
    a = 4 + false
}$
```

Results: ERROR: The [Expression] on line 4 col 20 is of type boolean which cannot be added to digits of type int

4.6 Tien Test

```
/* Thx Tien. */
{
    int a
    a = 0
    string z
    z = "bond"
    while (a != 9) {
       if (a != 5) {
           print("bond")
       }
       {
           a = 1 + a
           string b
           b = "james bond"
           print(b)
       }
    }
    {/*Holy Hell This is Disgusting*/}
    boolean c
    c = true
    boolean d
    d = (true == (true == false))
    d = (a == b)
    d = (1 == a)
    d = (1 != 1)
    d = ("string" == 1)
    d = (a != "string")
    d = ("string" != "string")
    if (d == true) {
        int c
        c = 1 + d
        if (c == 1) {
            print("ugh")
        }
    }
    while ("string" == a) {
        while (1 == true) {
            a = 1 + "string"
        }
    }
}$
```

Results: ERROR - Variable [b] on line 40 col 22 has not been previously declared.

4.7 Tien Boolean Hell

```
/* Thanks Tien. Assuming you get past Boolean Hell
 - there is a boolean being compared to
 - a string which will cause a type error */
{
    int a
    a = 4
    boolean b
    b = true
    boolean c
    string d
    d = "there is no spoon"
    c = (d != "there is a spoon")
    if(c == (false != (b == (true == (a == 3+1))))) {
        print((b != d))
    }
}$
```

Results: ERROR - The [Expression] on line 14 col 23 is of type boolean and is incompatibly compared to a type of string

5 Code Gen Fail Programs

5.1 Boolean Hell

```
/* This test case is included because it completely messed
 - up my AST with boolean hell and keeping track of boolexpr
 - may it serve as a good benchmark for those who come after
 - CREDIT: TIEN */
{
    int a
    a = 0
    boolean b
    b = false
    boolean c
    c = true
    while(((a!=9) == ("test" != "alan")) == ((5==5) != (b == c))) {
        print("a")
        string d
        d = "yes"
        print(d)
        {
            int a
            a = 5
        }
    }
}$
```

Results: ERROR: Please no boolean hell.

5.2 Max Memory

```
/* Valid code but can't fit into 256 bytes */
{
    int a
    int b
    int c
    int d
    a = 2
    {
        b = 5
        print(b)
        a = 1 + a
        {
            print(a)
            a = 5
        }
        if(a == b) {
            print("wowza")
        }
        int d
        d = 5
        {
            string d
            d = "hey"
            print(d)
            d = "sap"
            print(d)
        }
        print(d)
    }
    c = 4
    print(c)
    while (c != 7) {
        c = 1 + 1 + 1 + c
        print(c)
    }
```

```
            c = 9 + c
            print(c)
        }$
```

Results: ERROR: Max memory