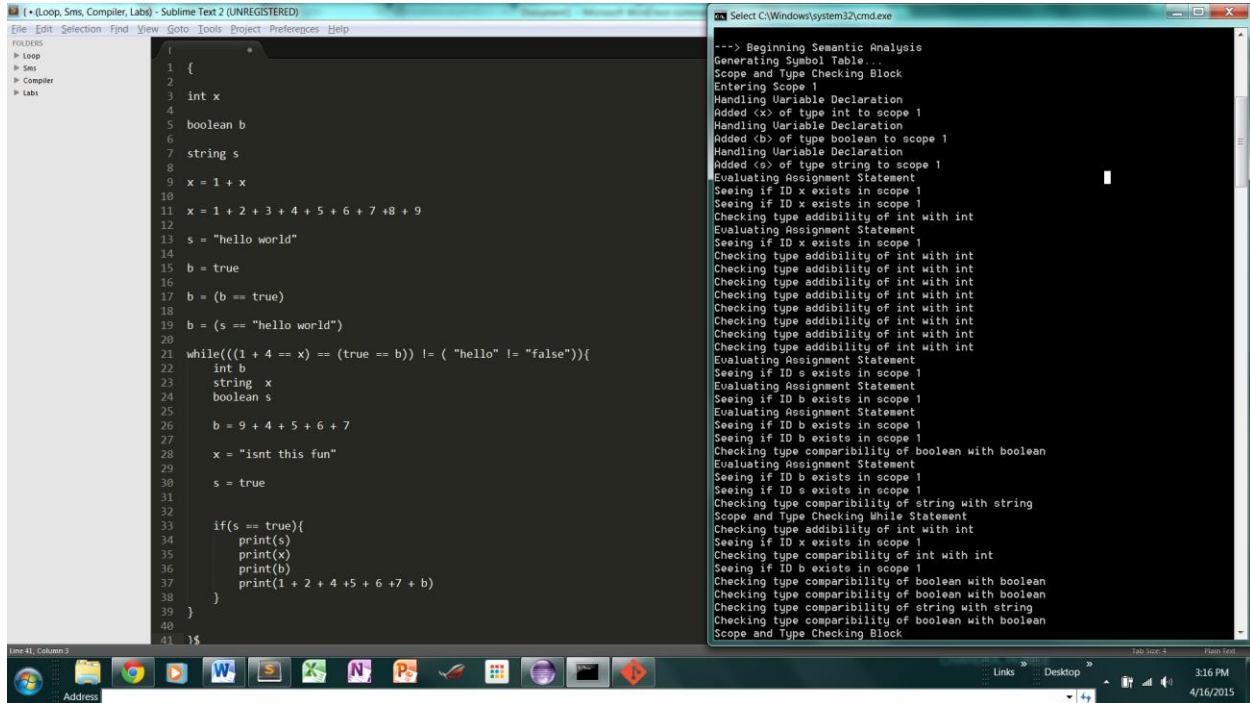


Andrew Langford

Project 2

TEST 1

This test's main purpose is to show the nesting of boolean and addition operations.



The screenshot displays a Sublime Text editor window with a C program and a terminal window showing its semantic analysis output.

C Program (Left Window):

```
1 {  
2  
3     int x  
4  
5     boolean b  
6  
7     string s  
8  
9     x = 1 + x  
10  
11    x = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9  
12  
13    s = "hello world"  
14  
15    b = true  
16  
17    b = (b == true)  
18  
19    b = (s == "hello world")  
20  
21    while(((1 + 4 == x) == (true == b)) != ("hello" != "false")){  
22        int b  
23        string x  
24        boolean s  
25  
26        b = 9 + 4 + 5 + 6 + 7  
27  
28        x = "isnt this fun"  
29  
30        s = true  
31  
32  
33        if(s == true){  
34            print(s)  
35            print(x)  
36            print(b)  
37            print(1 + 2 + 4 + 5 + 6 + 7 + b)  
38        }  
39    }  
40  
41 }
```

Semantic Analysis Output (Right Window):

```
---> Beginning Semantic Analysis  
Generating Symbol Table ...  
Scope and Type Checking Block  
Entering Scope 1  
Handling Variable Declaration  
Added (x) of type int to scope 1  
Handling Variable Declaration  
Added (b) of type boolean to scope 1  
Handling Variable Declaration  
Added (s) of type string to scope 1  
Evaluating Assignment Statement  
Seeing if ID x exists in scope 1  
Checking type addability of int with int  
Evaluating Assignment Statement  
Seeing if ID x exists in scope 1  
Checking type addability of int with int  
Checking type addability of int with int  
Checking type addability of int with int  
Checking type addability of int with int  
Checking type addability of int with int  
Checking type addability of int with int  
Checking type addability of int with int  
Evaluating Assignment Statement  
Seeing if ID s exists in scope 1  
Evaluating Assignment Statement  
Seeing if ID b exists in scope 1  
Evaluating Assignment Statement  
Seeing if ID b exists in scope 1  
Checking type comparability of boolean with boolean  
Evaluating Assignment Statement  
Seeing if ID b exists in scope 1  
Seeing if ID s exists in scope 1  
Checking type comparability of string with string  
Scope and Type Checking While Statement  
Checking type addability of int with int  
Seeing if ID x exists in scope 1  
Checking type comparability of int with int  
Seeing if ID b exists in scope 1  
Checking type comparability of boolean with boolean  
Checking type comparability of boolean with boolean  
Checking type comparability of string with string  
Checking type comparability of boolean with boolean  
Scope and Type Checking Block
```

The screenshot shows the Sublime Text 2 editor with a C program on the left and its compilation output on the right. The C program defines variables `int x`, `boolean b`, and `string s`, performs arithmetic on `x`, and uses `while` and `if` loops. The output window shows the compiler's internal checks, such as type comparability and scope management, and displays a Symbol Table Tree.

```
1 {
2
3 int x
4
5 boolean b
6
7 string s
8
9 x = 1 + x
10
11 x = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9
12
13 s = "hello world"
14
15 b = true
16
17 b = (b == true)
18
19 b = (s == "hello world")
20
21 while((1 + 4 == x) == (true == b)) != ( "hello" != "false")){
22     int b
23     string x
24     boolean s
25
26     b = 9 + 4 + 5 + 6 + 7
27
28     x = "isnt this fun"
29
30     s = true
31
32
33     if(s == true){
34         print(s)
35         print(x)
36         print(b)
37         print(1 + 2 + 4 + 5 + 6 + 7 + b)
38     }
39 }
40
41 }
```

Checking type comparability of boolean with boolean
Scope and Type Checking Block
Entering Scope 2
Handling Variable Declaration
Added (b) of type int to scope 2
Handling Variable Declaration
Added (x) of type string to scope 2
Handling Variable Declaration
Added (s) of type boolean to scope 2
Evaluating Assignment Statement
Seeing if ID b exists in scope 2
Checking type addibility of int with int
Checking type addibility of int with int
Checking type addibility of int with int
Checking type addibility of int with int
Evaluating Assignment Statement
Seeing if ID x exists in scope 2
Evaluating Assignment Statement
Seeing if ID s exists in scope 2
Scope and Type Checking If Statement
Seeing if ID s exists in scope 2
Checking type comparability of boolean with boolean
Scope and Type Checking Block
Entering Scope 3
Seeing if ID s exists in scope 3
checking parent scopes...
Seeing if ID x exists in scope 3
checking parent scopes...
Seeing if ID b exists in scope 3
checking parent scopes...
Seeing if ID b exists in scope 3
checking parent scopes...
Checking type addibility of int with int
Checking type addibility of int with int
Checking type addibility of int with int
Checking type addibility of int with int
Checking type addibility of int with int
Checking type addibility of int with int
---> Symbol Table Tree
Scope 1
string s
boolean b
int x
Scope 2
boolean s
int b
int b

This screenshot shows the same C program as the first image, but the output window displays a different set of compiler diagnostics. It includes a Symbol Table Hash and a more detailed Symbol Table Tree, showing the state of variables across different scopes and their initialization status.

```
1 {
2
3 int x
4
5 boolean b
6
7 string s
8
9 x = 1 + x
10
11 x = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9
12
13 s = "hello world"
14
15 b = true
16
17 b = (b == true)
18
19 b = (s == "hello world")
20
21 while((1 + 4 == x) == (true == b)) != ( "hello" != "false")){
22     int b
23     string x
24     boolean s
25
26     b = 9 + 4 + 5 + 6 + 7
27
28     x = "isnt this fun"
29
30     s = true
31
32
33     if(s == true){
34         print(s)
35         print(x)
36         print(b)
37         print(1 + 2 + 4 + 5 + 6 + 7 + b)
38     }
39 }
40
41 }
```

---> Symbol Table Tree
Scope 1
string s
boolean b
int x
Scope 2
boolean s
int b
string x
Scope 3
---> Symbol Table Hash
Identifier => s
Scope => 1
Type => string
isDeclared => true
lineDeclared => 7
isUsed => true
isInitialized => true

Identifier => b
Scope => 1
Type => boolean
isDeclared => true
lineDeclared => 5
isUsed => true
isInitialized => true

Identifier => x
Scope => 1
Type => int
isDeclared => true
lineDeclared => 3
isUsed => true
isInitialized => true

Identifier => s
Scope => 2
Type => boolean
isDeclared => true
lineDeclared => 24
isUsed => true
isInitialized => true

```
{
  {
    1 {
    2
    3 int x
    4
    5 boolean b
    6
    7 string s
    8
    9 x = 1 + x
    10
    11 x = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9
    12
    13 s = "hello world"
    14
    15 b = true
    16
    17 b = (b == true)
    18
    19 b = (s == "hello world")
    20
    21 while(((1 + 4 == x) == (true == b)) != ("hello" != "false")){
    22   int b
    23   string x
    24   boolean s
    25
    26   b = 9 + 4 + 5 + 6 + 7
    27
    28   x = "isnt this fun"
    29
    30   s = true
    31
    32   if(s == true){
    33     print(s)
    34     print(x)
    35     print(b)
    36     print(1 + 2 + 4 + 5 + 6 + 7 + b)
    37   }
    38 }
    39 }
    40 }
    41 }
```

```
isInitialized => true
identifier => b
Scope => 2
Type => int
isDeclared => true
lineDeclared => 22
isUsed => true
isInitialized => true

identifier => x
Scope => 2
Type => string
isDeclared => true
lineDeclared => 23
isUsed => true
isInitialized => true

---> Semantic Analysis Successful

---> Abstract SyntaxTree
Block
  VarDecl
    int
    x
  VarDecl
    boolean
    b
  VarDecl
    string
    s
  AssignmentStatement
    x
    +
    1
    x
  AssignmentStatement
    x
    +
    1
    +
    2
    +
```

TEST 2

The purpose of this test is to test scoping and operations across scopes.

```
1 {  
2   int x  
3   x = 4  
4  
5   string s  
6   s = "this suit"  
7  
8   boolean b  
9   b = false  
10  
11   {  
12     int y  
13     y = 5  
14  
15     string t  
16     t = "is black"  
17  
18     boolean c  
19     c = false  
20  
21     {  
22       int z  
23       z = 6  
24  
25       string u  
26       u = "not "  
27  
28       boolean d  
29       d = true  
30  
31       {  
32         z = 1 + y  
33         y = 2 + z  
34  
35         u = t  
36         t = s  
37  
38         d = (true == c)  
39  
40         c = (false != b)  
41       }  
42     }  
43   }  
44 }  
45  
46 }$
```

```
---> Beginning Semantic Analysis  
Generating Symbol Table...  
Scope and Type Checking Block  
Entering Scope 1  
Handling Variable Declaration  
Added (x) of type int to scope 1  
Evaluating Assignment Statement  
Seeing if ID x exists in scope 1  
Handling Variable Declaration  
Added (s) of type string to scope 1  
Evaluating Assignment Statement  
Seeing if ID s exists in scope 1  
Handling Variable Declaration  
Added (b) of type boolean to scope 1  
Evaluating Assignment Statement  
Seeing if ID b exists in scope 1  
Scope and Type Checking Block  
Entering Scope 2  
Handling Variable Declaration  
Added (y) of type int to scope 2  
Evaluating Assignment Statement  
Seeing if ID y exists in scope 2  
Handling Variable Declaration  
Added (t) of type string to scope 2  
Evaluating Assignment Statement  
Seeing if ID t exists in scope 2  
Handling Variable Declaration  
Added (c) of type boolean to scope 2  
Evaluating Assignment Statement  
Seeing if ID c exists in scope 2  
Scope and Type Checking Block  
Entering Scope 3  
Handling Variable Declaration  
Added (z) of type int to scope 3  
Evaluating Assignment Statement  
Seeing if ID z exists in scope 3  
Handling Variable Declaration  
Added (u) of type string to scope 3  
Evaluating Assignment Statement  
Seeing if ID u exists in scope 3  
Handling Variable Declaration  
Added (d) of type boolean to scope 3  
Evaluating Assignment Statement  
Seeing if ID d exists in scope 3  
Scope and Type Checking Block  
Entering Scope 4
```

```
1 {  
2   int x  
3   x = 4  
4  
5   string s  
6   s = "this suit"  
7  
8   boolean b  
9   b = false  
10  
11   {  
12     int y  
13     y = 5  
14  
15     string t  
16     t = "is black"  
17  
18     boolean c  
19     c = false  
20  
21     {  
22       int z  
23       z = 6  
24  
25       string u  
26       u = "not "  
27  
28       boolean d  
29       d = true  
30  
31       {  
32         z = 1 + y  
33         y = 2 + z  
34  
35         u = t  
36         t = s  
37  
38         d = (true == c)  
39  
40         c = (false != b)  
41       }  
42     }  
43   }  
44 }  
45  
46 }$
```

```
Scope and Type Checking Block  
Entering Scope 4  
Evaluating Assignment Statement  
Seeing if ID z exists in scope 4  
checking parent scopes...  
Seeing if ID y exists in scope 4  
checking parent scopes...  
checking parent scopes...  
Checking type addability of int with int  
Evaluating Assignment Statement  
Seeing if ID y exists in scope 4  
checking parent scopes...  
checking parent scopes...  
Seeing if ID z exists in scope 4  
checking parent scopes...  
Checking type addability of int with int  
Evaluating Assignment Statement  
Seeing if ID u exists in scope 4  
checking parent scopes...  
Seeing if ID t exists in scope 4  
checking parent scopes...  
Evaluating Assignment Statement  
Seeing if ID t exists in scope 4  
checking parent scopes...  
checking parent scopes...  
Seeing if ID a exists in scope 4  
checking parent scopes...  
checking parent scopes...  
Evaluating Assignment Statement  
Seeing if ID d exists in scope 4  
checking parent scopes...  
Seeing if ID c exists in scope 4  
checking parent scopes...  
Checking type comparability of boolean with boolean  
Evaluating Assignment Statement  
Seeing if ID c exists in scope 4  
checking parent scopes...  
checking parent scopes...  
Seeing if ID b exists in scope 4  
checking parent scopes...  
checking parent scopes...  
Checking type comparability of boolean with boolean
```

```
(• Loop, Sms, Compiler, Labs) - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  loop
  Sms
  Compiler
  Labs

1 {
2   int x
3   x = 4
4
5   string s
6   s = "this suit"
7
8   boolean b
9   b = false
10
11 {
12   int y
13   y = 5
14
15   string t
16   t = "is black"
17
18   boolean c
19   c = false
20
21 {
22   int z
23   z = 6
24
25   string u
26   u = "not"
27
28   boolean d
29   d = true
30
31 {
32   z = 1 + y
33   y = 2 + z
34
35   u = t
36   t = s
37
38   d = (true == c)
39
40   c = (false != b)
41 }
42 }
43 }
44 }
45 }
46 }$

C:\Windows\system32\cmd.exe
Checking type comparability of boolean with boolean
---> Symbol Table Tree
Scope 1
  boolean b
  string s
  int x
  Scope 2
    string t
    boolean c
    int y
    Scope 3
      boolean d
      string u
      int z
      Scope 4

---> Symbol Table Hash
Identifier => b
Scope => 1
Type => boolean
isDeclared => true
lineDeclared => 8
isUsed => true
isInitialized => true

Identifier => s
Scope => 1
Type => string
isDeclared => true
lineDeclared => 5
isUsed => true
isInitialized => true

Identifier => x
Scope => 1
Type => int
isDeclared => true
lineDeclared => 2
isUsed => true
isInitialized => true

Identifier => t
Scope => 2
```

```
(• Loop, Sms, Compiler, Labs) - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  loop
  Sms
  Compiler
  Labs

1 {
2   int x
3   x = 4
4
5   string s
6   s = "this suit"
7
8   boolean b
9   b = false
10
11 {
12   int y
13   y = 5
14
15   string t
16   t = "is black"
17
18   boolean c
19   c = false
20
21 {
22   int z
23   z = 6
24
25   string u
26   u = "not"
27
28   boolean d
29   d = true
30
31 {
32   z = 1 + y
33   y = 2 + z
34
35   u = t
36   t = s
37
38   d = (true == c)
39
40   c = (false != b)
41 }
42 }
43 }
44 }
45 }
46 }$

C:\Windows\system32\cmd.exe
Identifier => t
Scope => 2
Type => string
isDeclared => true
lineDeclared => 16
isUsed => true
isInitialized => true

Identifier => c
Scope => 2
Type => boolean
isDeclared => true
lineDeclared => 19
isUsed => true
isInitialized => true

Identifier => y
Scope => 2
Type => int
isDeclared => true
lineDeclared => 13
isUsed => true
isInitialized => true

Identifier => d
Scope => 3
Type => boolean
isDeclared => true
lineDeclared => 30
isUsed => true
isInitialized => true

Identifier => u
Scope => 3
Type => string
isDeclared => true
lineDeclared => 27
isUsed => true
isInitialized => true

Identifier => z
Scope => 3
```

```
1 {
2   int x
3   x = 4
4
5   string s
6   s = "this suit"
7
8   boolean b
9   b = false
10
11   {
12
13     int y
14     y = 5
15
16     string t
17     t = "is black"
18
19     boolean c
20     c = false
21
22     {
23       int z
24       z = 6
25
26       string u
27       u = "not "
28
29       boolean d
30       d = true
31
32       {
33         z = 1 + y
34         y = 2 + z
35
36         u = t
37         t = s
38
39         d = (true == c)
40         c = (false != b)
41       }
42     }
43   }
44 }
45
46 }
```

```
identifier => z
Scope      => 3
Type       => int
isDeclared => true
lineDeclared => 24
isUsed     => true
isInitialized => true

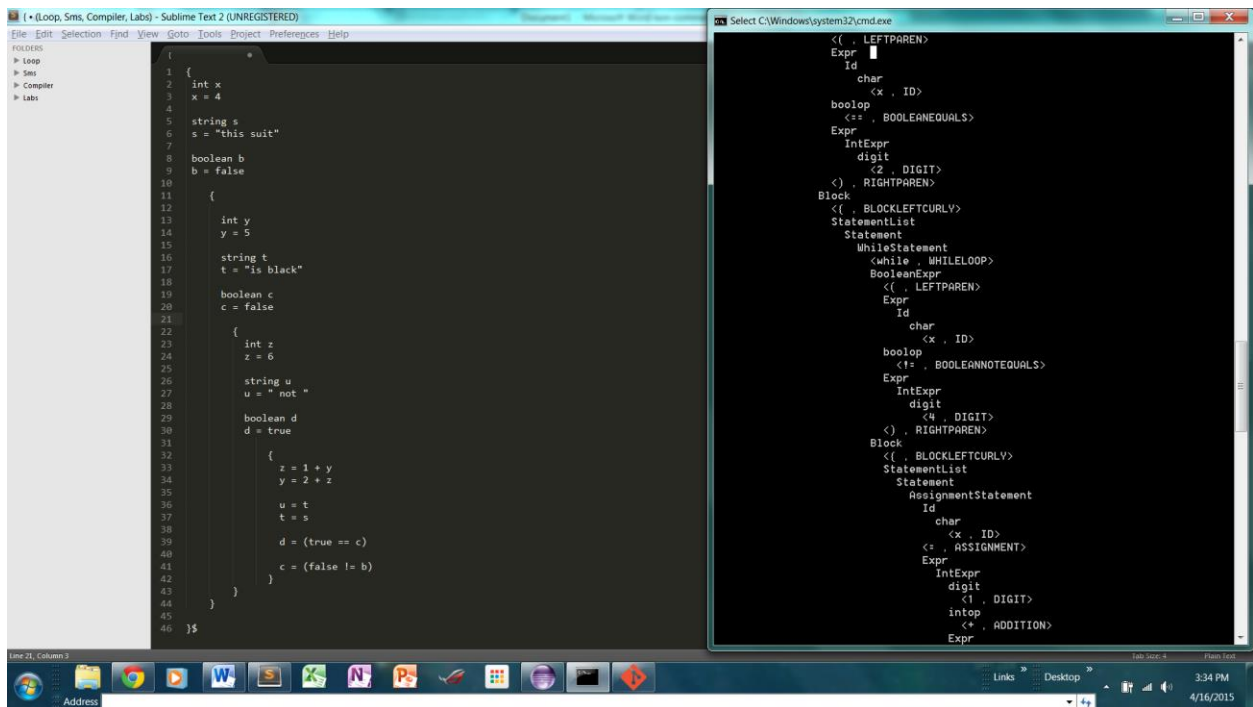
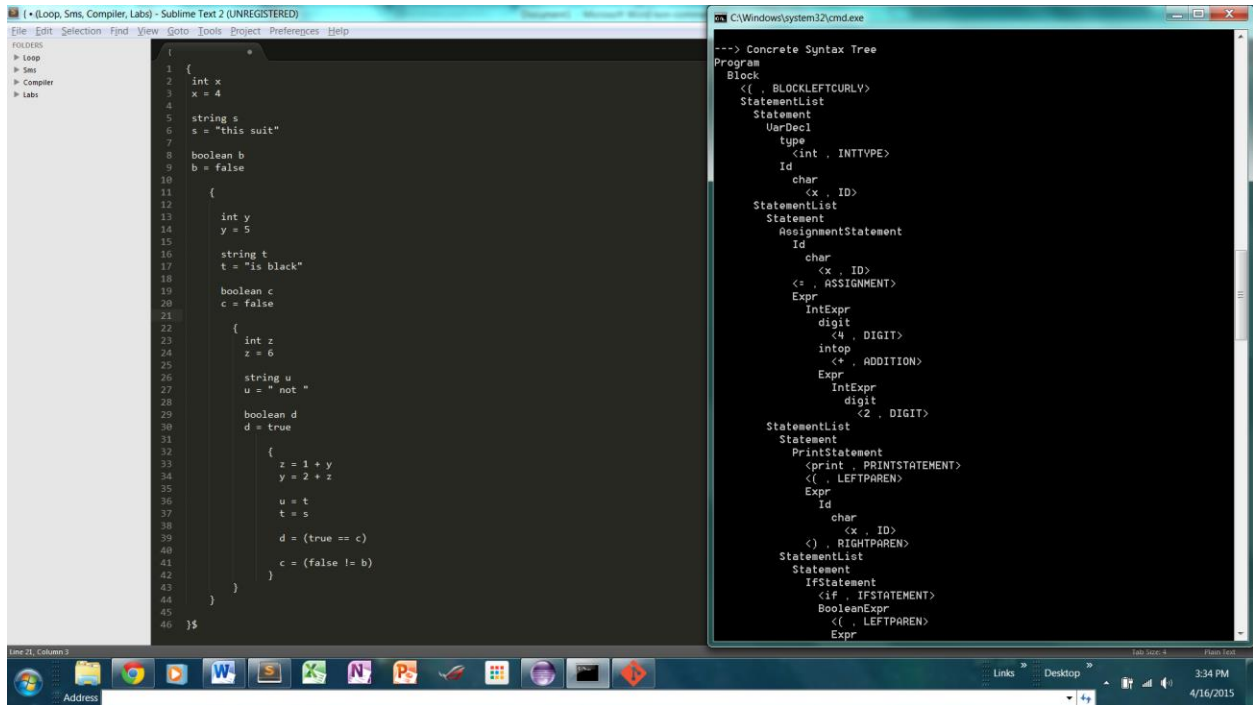
---> Semantic Analysis Successful

---> Abstract SyntaxTree
Block
  VarDecl
    int
    x
  AssignmentStatement
    x
    4
  VarDecl
    string
    s
  AssignmentStatement
    s
    "this suit"
  VarDecl
    boolean
    b
  AssignmentStatement
    b
    false
  Block
    VarDecl
      int
      y
    AssignmentStatement
      y
      5
    VarDecl
      string
      t
    AssignmentStatement
      t
      "is black"
    VarDecl
      boolean
      c
    AssignmentStatement
      c
      false
    VarDecl
      boolean
      d
    AssignmentStatement
      d
      true
    Block
      VarDecl
        int
        z
      AssignmentStatement
        z
        6
      VarDecl
        string
        u
      AssignmentStatement
        u
        "not "
      VarDecl
        boolean
        d
      AssignmentStatement
        d
        (true == c)
      VarDecl
        boolean
        c
      AssignmentStatement
        c
        (false != b)
    }
```

Test 3

Just a simple test to display the CST and AST:

A nice fluffy CST



The screenshot displays a compiler development environment with two windows. The left window, titled "Sublime Text 2 (UNREGISTERED)", contains a C program. The right window, titled "C:\Windows\system32\cmd.exe", shows the output of the compiler's semantic analysis, including the AST.

```
1 {  
2     int x  
3     x = 4  
4  
5     string s  
6     s = "this suit"  
7  
8     boolean b  
9     b = false  
10  
11     {  
12  
13         int y  
14         y = 5  
15  
16         string t  
17         t = "is black"  
18  
19         boolean c  
20         c = false  
21  
22         {  
23             int z  
24             z = 6  
25  
26             string u  
27             u = "not "  
28  
29             boolean d  
30             d = true  
31  
32             {  
33                 z = 1 + y  
34                 y = 2 + z  
35  
36                 u = t  
37                 t = s  
38  
39                 d = (true == c)  
40  
41                 c = (false != b)  
42             }  
43         }  
44     }  
45 }  
46 }
```

```
<+ . ADDITION>  
Expr  
Id  
char  
<x , ID>  
StatementList  
<() , BLOCKRIGHTCURLY>  
StatementList  
<() , BLOCKRIGHTCURLY>  
Statement  
PrintStatement  
<print , PRINTSTATEMENT>  
<() , LEFTPAREN>  
Expr  
Id  
char  
<x , ID>  
<() , RIGHTPAREN>  
StatementList  
<() , BLOCKRIGHTCURLY>  
<$ , EOF>  
  
---> Beginning Semantic Analysis  
Generating Symbol Table...  
Scope and Type Checking Block  
Entering Scope 1  
Handling Variable Declaration  
Added <x> of type int to scope 1  
Evaluating Assignment Statement  
Seeing if ID x exists in scope 1  
Checking type addability of int with int  
Seeing if ID x exists in scope 1  
Scope and Type Checking If Statement  
Seeing if ID x exists in scope 1  
Checking type comparability of int with int  
Scope and Type Checking Block  
Entering Scope 2  
Scope and Type Checking While Statement  
Seeing if ID x exists in scope 2  
checking parent scopes....  
Checking type comparability of int with int  
Scope and Type Checking Block  
Entering Scope 3  
Evaluating Assignment Statement  
Seeing if ID x exists in scope 3  
checking parent scopes....
```

A nice AST

The screenshot displays the same compiler development environment. The left window shows the C program, and the right window shows the output of the compiler's semantic analysis, including the AST.

```
1 {  
2     int x  
3     x = 4  
4  
5     string s  
6     s = "this suit"  
7  
8     boolean b  
9     b = false  
10  
11     {  
12  
13         int y  
14         y = 5  
15  
16         string t  
17         t = "is black"  
18  
19         boolean c  
20         c = false  
21  
22         {  
23             int z  
24             z = 6  
25  
26             string u  
27             u = "not "  
28  
29             boolean d  
30             d = true  
31  
32             {  
33                 z = 1 + y  
34                 y = 2 + z  
35  
36                 u = t  
37                 t = s  
38  
39                 d = (true == c)  
40  
41                 c = (false != b)  
42             }  
43         }  
44     }  
45 }  
46 }
```

```
Scope 2  
Scope 3  
---> Symbol Table Hash  
Identifier => x  
Scope => 1  
Type => int  
isDeclared => true  
lineDeclared => 2  
isUsed => true  
isInitialized => true  
  
---> Semantic Analysis Successful  
---> Abstract SyntaxTree  
Block  
VarDecl  
int  
x  
AssignmentStatement  
x  
+  
4  
2  
PrintStatement  
x  
IfStatement  
==  
x  
2  
Block  
WhileStatement  
!=  
x  
4  
Block  
AssignmentStatement  
x  
+  
1  
x  
PrintStatement  
x  
C:\Users\Andrew\workspace\Compiler\src
```


Test 4

Catch every semantical error/warning I can think of

The image shows a screenshot of a Windows-based IDE, likely Visual Studio Code, with two windows open. The left window displays a C program, and the right window shows the compiler's output.

Left Window (C Program):

```
1 // (Loop, Sms, Compiler, Labo) - Sublime Text 2 (UNREGISTERED)
2
3 #include <stdio.h>
4
5 int main()
6 {
7     int x;
8     string s;
9
10    x = 4 + 4;
11
12    boolean b;
13    b = (true == 4);
14    b = (true == "hello");
15
16    x = 1 + 2 + 3 + 4 + 5 + 6 + true;
17
18    int d;
19    d = 1 + 2 + 3 + 4 + "twiefuefwue";
20
21    x = 4 + d;
22
23    int g;
24    g = true;
25
26    boolean h;
27    h = "hello";
28
29    string t;
30    t = 3;
31
32    {
33        g = false;
34        t = true;
35        h = 5 + 7 + 8 + t;
36        int q;
37        string r;
38        boolean z;
39    }
40 }
```

Right Window (Compiler Output):

Warnings:

```
WARNING: [Line : 18] identifier d is used but not initialized
WARNING: [Line : 32] identifier t is used but not initialized
WARNING: [Line : 34] identifier r is never used.
WARNING: [Line : 33] identifier q is never used.
WARNING: [Line : 35] identifier z is never used.
```

Errors:

```
----> ERRORS
ERROR: [Line 4] variable <x> already declared for this scope
ERROR: [Line : 6] undeclared identifier <c>
ERROR: [Line : 9] Cannot compare boolean with int
ERROR: [Line : 10] Cannot compare boolean with string
ERROR: [Line : 12] Incompatible types. Expected: int Received: boolean
ERROR: [Line : 14] Incompatible types. Expected: int Received: string
ERROR: [Line : 21] Cannot assign boolean to id of type int
ERROR: [Line : 24] Cannot assign string to id of type boolean
ERROR: [Line : 27] Cannot assign int to id of type string
ERROR: [Line : 30] Cannot assign boolean to id of type int
ERROR: [Line : 31] Cannot assign boolean to id of type string
ERROR: [Line : 32] Incompatible types. Expected: int Received: string
ERROR: [Line : 32] Cannot assign int to id of type boolean
Semantic Analysis failed.....
Stopping Compilation!
```

The bottom of the image shows the Windows taskbar with various application icons and the system clock displaying 4/16/2015 at 3:58 PM.

Test 5

More errors

[illegible]

