Test 1:

{

int a

a = 5

}$

Kept from last tests. This was chosen to show basic variable declaration. It creates a simple enough concrete syntax tree, abstract syntax tree, and symbol table.

Test 2:

{

int a

a = 0

while (a != 9)

{

print (a)

a = 1 + a

}

}$this should give a warning

Kept from last time. It was then used to check the symbol table look ups that need to happen from child to parent scopes so that identifiers are correctly found. It shows that integer expression and Boolean expressions are formed correctly in the abstract syntax tree. This also shows the type checking between variables and primitive types.

Test 3:

{

boolean b

b = true

int z

if (b == true)

{

print ("b is equal to true")

z = 5

}

}

Kept from last time. This was chosen to show that variables can be assigned in different scopes. This also shows the type checking between primitives and variables. Shows that print can be used with string expressions correctly.

Test 4:

{

int a

{

print (a)

}

}$

Kept from last time. This was chosen to show the variable look up to the parent scope and to give the warning of using an uninitialized identifier.

Test 5:

{

int a

a = 5

boolean a

a = true

}$

This was chosen to show the error of identifiers being declared more than once in the same scope.

Test 6:

{

boolean a

a = ((true == true) != (false == (false == true)))

}$

This test was chosen to show that a correct abstract syntax tree is built for complicated Boolean expressions.

Test 7:

{

if ((5 != true) == true)

{

}

}$

This test was chosen to show that a Boolean expression can successfully type check its children.

Test 8:

{

{ b = true

{ boolean b } }

}$

This test was chosen to show that variables cannot be used without being declared first. Also, to show that the symbol table does correct look ups and does not check any children scopes for variables.

Test 9:

{

int a

boolean b

a = 1 + b

}$

This was chosen to show that only integer identifiers could be added in integer expressions. So, this fails when it tries to add b to 1.

Test 10:

{

int a

a = 0

boolean b

boolean c

while (((a != 9) == ("test" != "alan")) == ((5 == 5) != (b == c)))

{

print ("a")

string d

d = "yes"

print (d)

{ int a

a = 5 }

}

}$

This was chosen to show the warnings from declared but uninitialized variables, and to show that Boolean expression does the correct type checking. It was also chosen to show that variables can be declared again in a child scope properly. The test also shows that the symbol table keeps track of variables in their correct scope.