Test 1:

{

int a

a = 5

}$

This was chosen to show basic variable declaration. It creates a simple enough concrete syntax tree and symbol table to check that things are working without going to some of the depth that some of the other programs do. I also mean literal depth of the tree.

Test 2:

{

int a

a = 0

while (a != 9)

{

print (a)

a = 1 + a

}

}$this should give a warning

This was chosen first of all for the warning that it would give since there is information after the end of file. 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. Then, it was just basic testing of parsing while and addition statements to make sure that they are correct even inside other blocks.

Test 3:

{

boolean b

b = true

int z

if (b == true)

{

print ("b is equal to true")

z = 5

}

}

This was chosen to show that variables can assigned in different scopes. This is different from the previous one because you can walk through the checks in the previous one to make sure it is checking inside the while loop, but this program shows that without having to do any extra work. It will also give a warning that there is no end of file and that one is being added. Then, the program checks that the print statement works with a string expression and that the if format works correctly.

Test 4:

{

int a

{

print (a)

}

}$

This was chosen to give the error that the variable being used was never declared. So, the program will get to that point in parse then error out.

Test 5:

{

int a

a = ^

}$

This was chosen to give a lexical error because ^ is not a recognized symbol in our language. It will reach that point and error out.

Test 6:

{

string a

a = "abc"

(

)

}$

This test was chosen to show the parse error that would be displayed if a person used parentheses instead of curly braces to try and declare a new scope. It will reach that point in parse then error out.

Test 7:

{

if ((5 !=6) == true)

{

}

}$

This test was chosen that a Boolean expression can correctly recursively call other expressions that are Boolean expressions.

Test 8:

{

string nn

nn = "test"

}$

This test was chosen the fact the only a single character may be used to represent identifiers. This will fail at parse because white space is ignored, so it will look for an assignment on the second n and error out.

Test 9:

{

5 + 6

}$

This was chosen to show that in a block expressions cannot be directly parsed. The program must first use a statement. The program will error out in parse when it is looking for a statement.

Test 10:

{

int a

a = 0

while (a = 9)

{

}

}$

This chosen both to test that Boolean expressions would correctly fail but also what would happen if someone forgot an equal sign in a double equal. The program will error out on parse when it recognizes there is an assignment statement instead of a Boolean expression.