Name: ***Ben Puryear***

Assignment: 7

Correct and Complete:	24 / 24
Evidence and Quality of Testing:	2 / 2
Clean Code:	2 / 2
Writeup:	2 / 2
Total Score:	30 / 30

Comments:

1. Good job!

Test output:

```
1. RUNNING: ./code_generator_tests
[==========] Running 53 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 53 tests from BasicCodeGenTest
[ RUN      ] BasicCodeGenTest.EmptyProram
[       OK ] BasicCodeGenTest.EmptyProram (0 ms)
[ RUN      ] BasicCodeGenTest.SimplePrint
[       OK ] BasicCodeGenTest.SimplePrint (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleVariableDecls
[       OK ] BasicCodeGenTest.SimpleVariableDecls (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleVariableAssigns
[       OK ] BasicCodeGenTest.SimpleVariableAssigns (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleAdd
[       OK ] BasicCodeGenTest.SimpleAdd (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleSub
[       OK ] BasicCodeGenTest.SimpleSub (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleMult
[       OK ] BasicCodeGenTest.SimpleMult (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleDiv
[       OK ] BasicCodeGenTest.SimpleDiv (0 ms)
[ RUN      ] BasicCodeGenTest.LongerArithExpr
[       OK ] BasicCodeGenTest.LongerArithExpr (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleAnd
[       OK ] BasicCodeGenTest.SimpleAnd (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleOr
[       OK ] BasicCodeGenTest.SimpleOr (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleNot
[       OK ] BasicCodeGenTest.SimpleNot (0 ms)
[ RUN      ] BasicCodeGenTest.LongerBoolExpr
[       OK ] BasicCodeGenTest.LongerBoolExpr (0 ms)
[ RUN      ] BasicCodeGenTest.TrueNumberComparisons
[       OK ] BasicCodeGenTest.TrueNumberComparisons (0 ms)
[ RUN      ] BasicCodeGenTest.FalseNumberComparisons
[       OK ] BasicCodeGenTest.FalseNumberComparisons (0 ms)
[ RUN      ] BasicCodeGenTest.TrueAlphaComparisons
[       OK ] BasicCodeGenTest.TrueAlphaComparisons (0 ms)
[ RUN      ] BasicCodeGenTest.FalseAlphaComparisons
[       OK ] BasicCodeGenTest.FalseAlphaComparisons (0 ms)
```

```
[ RUN      ] BasicCodeGenTest.NullComparisons
[       OK ] BasicCodeGenTest.NullComparisons (1 ms)
[ RUN      ] BasicCodeGenTest.BasicWhile
[       OK ] BasicCodeGenTest.BasicWhile (0 ms)
[ RUN      ] BasicCodeGenTest.MoreInvolvedWhile
[       OK ] BasicCodeGenTest.MoreInvolvedWhile (0 ms)
[ RUN      ] BasicCodeGenTest.NestedWhile
[       OK ] BasicCodeGenTest.NestedWhile (0 ms)
[ RUN      ] BasicCodeGenTest.BasicFor
[       OK ] BasicCodeGenTest.BasicFor (0 ms)
[ RUN      ] BasicCodeGenTest.NestedFor
[       OK ] BasicCodeGenTest.NestedFor (0 ms)
[ RUN      ] BasicCodeGenTest.ForOuterShadowing
[       OK ] BasicCodeGenTest.ForOuterShadowing (0 ms)
[ RUN      ] BasicCodeGenTest.ForInnerShadowing
[       OK ] BasicCodeGenTest.ForInnerShadowing (0 ms)
[ RUN      ] BasicCodeGenTest.BasicIf
[       OK ] BasicCodeGenTest.BasicIf (0 ms)
[ RUN      ] BasicCodeGenTest.ConsecutiveIfs
[       OK ] BasicCodeGenTest.ConsecutiveIfs (0 ms)
[ RUN      ] BasicCodeGenTest.BasicElseIf
[       OK ] BasicCodeGenTest.BasicElseIf (0 ms)
[ RUN      ] BasicCodeGenTest.NoArgCall
[       OK ] BasicCodeGenTest.NoArgCall (0 ms)
[ RUN      ] BasicCodeGenTest.OneArgCall
[       OK ] BasicCodeGenTest.OneArgCall (0 ms)
[ RUN      ] BasicCodeGenTest.TwoArgCall
[       OK ] BasicCodeGenTest.TwoArgCall (0 ms)
[ RUN      ] BasicCodeGenTest.ThreeArgCall
[       OK ] BasicCodeGenTest.ThreeArgCall (0 ms)
[ RUN      ] BasicCodeGenTest.MultilevelCall
[       OK ] BasicCodeGenTest.MultilevelCall (0 ms)
[ RUN      ] BasicCodeGenTest.BasicRecursion
[       OK ] BasicCodeGenTest.BasicRecursion (0 ms)
[ RUN      ] BasicCodeGenTest.EmptyStruct
[       OK ] BasicCodeGenTest.EmptyStruct (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleFieldStruct
[       OK ] BasicCodeGenTest.SimpleFieldStruct (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleTwoFieldStruct
[       OK ] BasicCodeGenTest.SimpleTwoFieldStruct (0 ms)
[ RUN      ] BasicCodeGenTest.FieldInitialization
[       OK ] BasicCodeGenTest.FieldInitialization (0 ms)
[ RUN      ] BasicCodeGenTest.StructAssign
[       OK ] BasicCodeGenTest.StructAssign (0 ms)
[ RUN      ] BasicCodeGenTest.TwoStructs
[       OK ] BasicCodeGenTest.TwoStructs (0 ms)
[ RUN      ] BasicCodeGenTest.RecursiveStruct
[       OK ] BasicCodeGenTest.RecursiveStruct (0 ms)
[ RUN      ] BasicCodeGenTest.StructsAsParams
[       OK ] BasicCodeGenTest.StructsAsParams (0 ms)
[ RUN      ] BasicCodeGenTest.ArrayCreation
[       OK ] BasicCodeGenTest.ArrayCreation (0 ms)
[ RUN      ] BasicCodeGenTest.ArrayInit
[       OK ] BasicCodeGenTest.ArrayInit (0 ms)
[ RUN      ] BasicCodeGenTest.ArrayOfStruct
[       OK ] BasicCodeGenTest.ArrayOfStruct (0 ms)
[ RUN      ] BasicCodeGenTest.StructOfArray
[       OK ] BasicCodeGenTest.StructOfArray (0 ms)
[ RUN      ] BasicCodeGenTest.ArrayAsParam
[       OK ] BasicCodeGenTest.ArrayAsParam (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleToStr
[       OK ] BasicCodeGenTest.SimpleToStr (0 ms)
```

```
[ RUN      ] BasicCodeGenTest.SimpleToInt
[       OK ] BasicCodeGenTest.SimpleToInt (0 ms)
[ RUN      ] BasicCodeGenTest.SimpleToDouble
[       OK ] BasicCodeGenTest.SimpleToDouble (0 ms)
[ RUN      ] BasicCodeGenTest.StringLength
[       OK ] BasicCodeGenTest.StringLength (0 ms)
[ RUN      ] BasicCodeGenTest.StringGet
[       OK ] BasicCodeGenTest.StringGet (0 ms)
[ RUN      ] BasicCodeGenTest.StringConcat
[       OK ] BasicCodeGenTest.StringConcat (0 ms)
[----------] 53 tests from BasicCodeGenTest (19 ms total)

[----------] Global test environment tear-down
[==========] 53 tests from 1 test suite ran. (19 ms total)
[  PASSED  ] 53 tests.

2. RUNNING: ./mypl examples/exec-arrays-structs.mypl
should be 20: 20
should be 10: 10
should be 30: 30
should be 5: 5
should be 15: 15

3. RUNNING: ./mypl examples/exec-arrays.mypl
should be [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
should be [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
should be [9, 10, 7, 8, 5, 6, 3, 4, 1, 2]: [9, 10, 7, 8, 5, 6, 3, 4, 1, 2]
should be [9, 10, 7, 8, 5, 6, 3, 4, 1, 2]: [9, 10, 7, 8, 5, 6, 3, 4, 1, 2]

4. RUNNING: ./mypl examples/exec-basic-function.mypl
... in f1
Should be 7: 7
... in f2, x = ab
... in f3, after f2, x = abab
Should be abab: abab

5. RUNNING: ./mypl examples/exec-built-ins.mypl <<< hello
... in f1
Should be 7: 7
... in f2, x = ab
... in f3, after f2, x = abab
Should be abab: abab

6. RUNNING: ./mypl examples/exec-catalan-nums.mypl
Catalan number 0 = 1
Catalan number 1 = 1
Catalan number 2 = 2
Catalan number 3 = 5
Catalan number 4 = 14
Catalan number 5 = 42
Catalan number 6 = 132

7. RUNNING: ./mypl examples/exec-cond.mypl
Should be 0: 0
Should print else case: else case
Should print elseif case: elseif case
Should print else case: else case
Should print oops: oops
should be 1 2 ... 6: 1 2 3 4 5 6
should be 5 4 ... 0: 5 4 3 2 1 0

8. RUNNING: ./mypl examples/exec-expr.mypl
```

```
String Tests:
  Should be true 'abc' < 'abd': true
  Should be true 'abc' <= 'abd': true
  Should be true 'abd' > 'abc': true
  Should be true 'abc' >= 'abc': true
  Should be true 'abc == 'abc': true
  Should be true 'abd != 'abc': true
Integer Tests:
  Should be '5': 5
  Should be '9': 9
  Should be '6': 6
  Should be '6': 6
  Should be '1': 1
  Should be '2': 2
  Should be '-1': -1
  Should be true 3 < 4: true
  Should be true 3 <= 4: true
  Should be true 4 > 3: true
  Should be true 4 >= 3: true
  Should be true 4 == 4: true
  Should be true 4 != 3: true
  Should be true not 4 != 4: true
Double Tests:
  Should be '5.500000': 5.500000
  Should be '9.250000': 9.250000
  Should be '6.750000': 6.750000
  Should be '9.375000': 9.375000
  Should be '1.750000': 1.750000
  Should be '2.080000': 2.080000
  Should be '-3.400000': -3.400000
  Should be true 3.1 < 4.2: true
  Should be true 3.1 <= 4.2: true
  Should be true 4.2 > 3.1: true
  Should be true 4.2 >= 3.1: true
  Should be true 4.2 == 4.2: true
  Should be true 4.2 != 3.1: true
Bool Tests:
  Should be true (not false): true
  Should be true (true and true): true
  Should be true (not false and true): true
  Should be true ((not false) and true): true
  Should be true (not (true and false)): false
  Should be true (true or false): true
  Should be true (false or true): true
  Should be true (false or (not false)): true
  Should be true (not false or false): true
Char Tests:
  Should be true 'a' < 'b': true
  Should be true 'a' <= 'a': true
  Should be true 'd' > 'c': true
  Should be true 'b' >= 'a': true
  Should be true 'a' == 'a': true
  Should be true 'b' != 'a': true

9. RUNNING: ./mypl examples/exec-fac.mypl
the factorial of 12 is 479001600

10. RUNNING: ./mypl examples/exec-fib.mypl
fib(0) = 0
fib(1) = 1
fib(2) = 1
fib(3) = 2
```

```
fib(4) = 3
fib(5) = 5
fib(6) = 8
fib(7) = 13
fib(8) = 21
fib(9) = 34
fib(10) = 55
fib(11) = 89
fib(12) = 144
fib(13) = 233
fib(14) = 377
fib(15) = 610
fib(16) = 987
fib(17) = 1597
fib(18) = 2584
fib(19) = 4181
fib(20) = 6765
fib(21) = 10946
fib(22) = 17711
fib(23) = 28657
fib(24) = 46368
fib(25) = 75025

11. RUNNING: ./mypl examples/exec-hello.mypl
Hello World!

12. RUNNING: ./mypl examples/exec-linked-list.mypl
[10, 20, 30, 40, 50]

13. RUNNING: ./mypl examples/exec-more-structs.mypl
Should be 0: 0
Should be 1: 1
Should be 5: 5
Should be 3: 3

14. RUNNING: ./mypl examples/exec-nested-if.mypl
test 1: pass
test 2: pass
test 3: pass

15. RUNNING: ./mypl examples/exec-simple-struct.mypl
t1.x should be 0: 0
t1.y should be 1: 1
t1.x should now be 5: 5
t1.y should now be 6: 6
t1.x should now be 7: 7
t1.y should now be 8: 8

17. RUNNING: ./mypl examples/exec-tree.mypl
Tree Values: 1 2 5 7 10 12 13 14 15
Tree Height: 5

18. RUNNING: ./mypl examples/exec-while.mypl
1, 1, 1
1, 1, 2
1, 1, 3
1, 2, 1
1, 2, 2
1, 2, 3
1, 3, 1
1, 3, 2
1, 3, 3
```

```
2, 1, 1
2, 1, 2
2, 1, 3
2, 2, 1
2, 2, 2
2, 2, 3
2, 3, 1
2, 3, 2
2, 3, 3
3, 1, 1
3, 1, 2
3, 1, 3
3, 2, 1
3, 2, 2
3, 2, 3
3, 3, 1
3, 3, 2
3, 3, 3
```