Name: *Dominic MacIsaac*

Assignment: 7

Complete & Correct:	40 / 40
Tests:	0 / 0
Format and Comments:	5 / 5
Writeup:	5 / 5
Total Score:	40 / 50

Comments:

1. good job!
2. great example programs
3. (-10) 1-week late penalty

Unit test output:

```
Loading:
Loading: 0 packages loaded
Analyzing: target //:code-generator-test (0 packages loaded, 0 targets configured)
INFO: Analyzed target //:code-generator-test (2 packages loaded, 140 targets configured).
INFO: Found 1 test target...
[1 / 6] [Prepa] Expanding template code-generator-test
[7 / 8] [Prepa] Building code-generator-test.jar (1 source file)
INFO: From Testing //:code-generator-test:
==================== Test output for //:code-generator-test:
JUnit4 Test Runner
.....................................
Time: 0.073

OK (42 tests)


BazelTestRunner exiting with a return value of 0
JVM shutdown hooks (if any) will run now.
The JVM will exit once they complete.

-- JVM shutdown starting at 2022-05-04 16:19:03 --

================================================================================
Target //:code-generator-test up-to-date:
  bazel-bin/code-generator-test.jar
  bazel-bin/code-generator-test
INFO: Elapsed time: 2.606s, Critical Path: 2.06s
INFO: 9 processes: 4 internal, 3 linux-sandbox, 2 worker.
INFO: Build completed successfully, 9 total actions
//:code-generator-test                                          PASSED in 0.4s

Executed 1 out of 1 test: 1 test passes.
There were tests whose specified size is too big. Use the --test_verbose_timeout_warnings command line option to see
     which ones these are.
INFO: Build completed successfully, 9 total actions
```

Program test output:

```
>>> bazel-bin/mypl ../examples-orig/exec-hello.mypl
Hello World!

>>> bazel-bin/mypl ../examples-orig/exec-expr.mypl
String Tests:
  Should be 'a': a
  Should be 'a': a
  Should be 'a': a
  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 '0': 0
  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 '5.5': 5.5
  Should be '9.25': 9.25
  Should be '6.75': 6.75
  Should be '9.375': 9.375
  Should be '1.75': 1.75
  Should be '2.08': 2.08
  Should be '-3.4': -3.4
  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)): true
  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

>>> bazel-bin/mypl ../examples-orig/exec-basic-function.mypl
... in f1
Should be 7: 7
... in f2, x = ab
... in f3, after f2, x = abab
Should be abab: abab

>>> bazel-bin/mypl ../examples-orig/exec-cond.mypl
Should be 0: 0
Should print else case: else case
```

```
Should print elif case: elif 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

>>> bazel-bin/mypl ../examples-orig/exec-nested-if.mypl
test 1: pass
test 2: pass
test 3: pass

>>> bazel-bin/mypl ../examples-orig/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

>>> bazel-bin/mypl ../examples-orig/exec-fac.mypl
The factorial of 12 should be 479001600: 479001600

>>> bazel-bin/mypl ../examples-orig/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

>>> bazel-bin/mypl ../examples-orig/exec-simple-udt.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

>>> bazel-bin/mypl ../examples-orig/exec-more-udt.mypl
Should be 0: 0
Should be 1: 1
Should be 5: 5
Should be 3: 3

>>> bazel-bin/mypl ../examples-orig/exec-linked-list.mypl
[10, 20, 30, 40, 50]

>>> bazel-bin/mypl ../examples-orig/exec-tree.mypl
Tree Values: 1 2 5 7 10 12 13 14 15
Tree Height: 5


>>> Done
```