```
print""
print "Megan Mulcahy's Test suite for LOX Interpreter"
print "+-----------------------------------------+"
print "Assignment:"

var a = "a";
var b = "b";
print a;        // a
print b;        // b
a = b;
print a;        // b
print "";

var a = "before";
print a;        // before
a = "after";
print a;        // after
print a = "arg"; // arg

print "";

{
  var a = "before";
  print a;      // expect: before

  a = "after";
  print a;      // expect: after

  print a = "arg"; // expect: arg
  print a;      // expect: arg
}

print ""

var a = "before";
var c = a = "var";
print a;        // expect: var
print c;        // expect: var

print ""
print "+-----------------------------------------+"
print "Empty Blocks:"

if (true) {}            //expect nothing
if (false) {} else {}   //expect nothing


print ""
print "+-----------------------------------------+"
print "Inner & Outer Blocks:"
```

```
var a = "outer";
{
  var a = "inner";
  print a;          // expect: inner
}
print a;         // expect: outer


print ""
print "+----------------------------------------+"
print "Equality"

print true == true;    // expect: true
print true == false;   // expect: false
print false == true;   // expect: false
print false == false;  // expect: true
print ""

print true == 1;        // expect: false
print false == 0;       // expect: false
print true == "true";   // expect: false
print false == "false"; // expect: false
print false == "";      // expect: false
print ""

print true != true;    // expect: false
print true != false;   // expect: true
print false != true;   // expect: true
print false != false;  // expect: false
print ""

// Not equal to other types.
print true != 1;        // expect: true
print false != 0;       // expect: true
print true != "true";   // expect: true
print false != "false"; // expect: true
print false != "";      // expect: true
print ""

print !true;            // expect: false
print !false;           // expect: true
print !!true;           // expect: true

print ""
print "+----------------------------------------+"
print "Closures:"

var f;
var g;
```

```
{
  var local = "local";
  fun f_() {
    print local;
    local = "after f";
    print local;
  }
  f = f_;

  fun g_() {
    print local;
    local = "after g";
    print local;
  }
  g = g_;
}

f();
// expect: local
// expect: after f

g();
// expect: after f
// expect: after g

print ""
var f;

fun foo(param) {
  fun f_() {
    print param;
  }
  f = f_;
}
foo("param");

f();                // expect: param

print ""
var f;

{
  var local = "local";
  fun f_() {
    print local;
  }
  f = f_;
}
```

```
f();            // expect: local

print ""

var f;

fun f1() {
  var a = "a";
  fun f2() {
    var b = "b";
    fun f3() {
      var c = "c";
      fun f4() {
        print a;
        print b;
        print c;
      }
      f = f4;
    }
    f3();
  }
  f2();
}
f1();

f();
                // expect: a
                // expect: b
                // expect: c

print ""
print "+-----------------------------------------+"
print "Expressions:"

print (5 - (3 - 1)) + -1        //expect: 2

print ""
print "+-----------------------------------------+"
print "For:"

var f1;
var f2;
var f3;

for (var i = 1; i < 4; i = i + 1) {
  var j = i;
  fun f() {
    print i;
    print j;
  }
```

```
  if (j == 1) f1 = f;
  else if (j == 2) f2 = f;
  else f3 = f;
}

f1();               // expect: 4
                    // expect: 1
f2();               // expect: 4
                    // expect: 2
f3();               // expect: 4
                    // expect: 3

print ""

fun f() {
  for (;;) {
    var i = "i";
    fun g() { print i; }
    return g;
  }
}

var h = f();
h(); // expect: i

print ""

fun f() {
  for (;;) {
    var i = "i";
    return i;
  }
}

print f();
// expect: i

print ""

// Single-expression body.
for (var c = 0; c < 3;) print c = c + 1;
                // expect: 1
                // expect: 2
                // expect: 3

print ""

// Block body.
for (var a = 0; a < 3; a = a + 1) {
```

```
  print a;
}
                // expect: 0
                // expect: 1
                // expect: 2

print ""

// No clauses.
fun foo() {
  for (;;) return "done";
}
print foo();    // expect: done

print ""

// No variable.
var i = 0;
for (; i < 2; i = i + 1) print i;
                // expect: 0
                // expect: 1

print ""

// No condition.
fun bar() {
  for (var i = 0;; i = i + 1) {
    print i;
    if (i >= 2) return;
  }
}
bar();
                // expect: 0
                // expect: 1
                // expect: 2

print ""

// No increment.
for (var i = 0; i < 2;) {
  print i;
  i = i + 1;
}
                // expect: 0
                // expect: 1

print ""

// Statement bodies.
for (; false;) if (true) 1; else 2;
```

```
for (; false;) while (true) 1;
for (; false;) for (;;) 1;

print ""
print "+----------------------------------------+"
print "Functions:"

fun f() {}
print f(); // expect: nil

print ""

{
  fun fib(n) {
    if (n < 2) return n;
    return fib(n - 1) + fib(n - 2);
  }

  print fib(8);         // expect: 21
}

print ""

fun isEven(n) {
  if (n == 0) return true;
  return isOdd(n - 1);
}

fun isOdd(n) {
  if (n == 0) return false;
  return isEven(n - 1);
}

print isEven(4);        // expect: true
print isOdd(3);         // expect: true

print ""

fun returnArg(arg) {
  return arg;
}

fun returnFunCallWithArg(func, arg) {
  return returnArg(func)(arg);
}

fun printArg(arg) {
  print arg;
}
```

```
returnFunCallWithArg(printArg, "hello world"); // expect: hello world

print ""

fun f0() { return 0; }
print f0(); // expect: 0

fun f1(a) { return a; }
print f1(1); // expect: 1

fun f2(a, b) { return a + b; }
print f2(1, 2); // expect: 3

fun f3(a, b, c) { return a + b + c; }
print f3(1, 2, 3); // expect: 6

fun f4(a, b, c, d) { return a + b + c + d; }
print f4(1, 2, 3, 4); // expect: 10

fun f5(a, b, c, d, e) { return a + b + c + d + e; }
print f5(1, 2, 3, 4, 5); // expect: 15

fun f6(a, b, c, d, e, f) { return a + b + c + d + e + f; }
print f6(1, 2, 3, 4, 5, 6); // expect: 21

fun f7(a, b, c, d, e, f, g) { return a + b + c + d + e + f + g; }
print f7(1, 2, 3, 4, 5, 6, 7); // expect: 28

fun f8(a, b, c, d, e, f, g, h) { return a + b + c + d + e + f + g +
h; }
print f8(1, 2, 3, 4, 5, 6, 7, 8); // expect: 36

print ""

fun foo() {}
print foo; // expect: <fn foo>

print clock; // expect: <native fn>

print ""
print "Recursion"

fun fib(n) {
  if (n < 2) return n;
  return fib(n - 1) + fib(n - 2);
}

print fib(8); // expect: 21

print ""
```

```
print "+----------------------------------------+"
print "If:"

if (true) if (false) print "bad"; else print "good"; // expect: good
if (false) if (true) print "bad"; else print "bad";

print ""

if (true) print "good"; else print "bad"; // expect: good
if (false) print "bad"; else print "good"; // expect: good

// Allow block body.
if (false) nil; else { print "block"; } // expect: block

print ""

if (true) print "good"; // expect: good
if (false) print "bad";

print ""

// Allow block body.
if (true) { print "block"; } // expect: block

print ""

// Assignment in if condition.
var a = false;
if (a = true) print a; // expect: true

print ""

// False and nil are false.
if (false) print "bad"; else print "false"; // expect: false
if (nil) print "bad"; else print "nil"; // expect: nil

print ""

// Everything else is true.
if (true) print true; // expect: true
if (0) print 0; // expect: 0
if ("") print "empty"; // expect: empty

print ""
print "+----------------------------------------+"
print "And:"

print false and 1;          // expect: false
print true and 1;           // expect: 1
print 1 and 2 and false;    // expect: false
```

```
print ""

print 1 and true; // expect: true
print 1 and 2 and 3; // expect: 3

print ""

// Short-circuit at the first false argument.
var a = "before";
var b = "before";
(a = true) and
    (b = false) and
    (a = "bad");
print a; // expect: true
print b; // expect: false

print ""
print "+----------------------------------------+"
print "Or:"

print 1 or true;                // expect: 1
print false or 1;               // expect: 1
print false or false or true;   // expect: true

print ""
// Return the last argument if all are false.
print false or false;           // expect: false
print false or false or false;  // expect: false

print ""
// Short-circuit at the first true argument.
var a = "before";
var b = "before";
(a = false) or
    (b = true) or
    (a = "bad");
print a;        // expect: false
print b;        // expect: true

print ""
print "+----------------------------------------+"
print "Numbers:"

print 123;     // expect: 123
print 987654;  // expect: 987654
print 0;       // expect: 0
print -0;      // expect: 0
print 123.456; // expect: 123.456
print -0.001;  // expect: -0.001
```

```
print ""
print "+-----------------------------------------+"
print "Add & Subtract:"

print 123 + 456;        // expect: 579

print ""

print 4 - 3; // expect: 1
print 1.2 - 1.2; // expect: 0

print ""
print "+-----------------------------------------+"
print "Comparisons:"

print 1 < 2;    // expect: true
print 2 < 2;    // expect: false
print 2 < 1;    // expect: false
print ""

print 1 <= 2;    // expect: true
print 2 <= 2;    // expect: true
print 2 <= 1;    // expect: false
print ""

print 1 > 2;    // expect: false
print 2 > 2;    // expect: false
print 2 > 1;    // expect: true
print ""

print 1 >= 2;    // expect: false
print 2 >= 2;    // expect: true
print 2 >= 1;    // expect: true
print ""

// Zero and negative zero compare the same.
print 0 < -0; // expect: false
print -0 < 0; // expect: false
print 0 > -0; // expect: false
print -0 > 0; // expect: false
print 0 <= -0; // expect: true
print -0 <= 0; // expect: true
print 0 >= -0; // expect: true
print -0 >= 0; // expect: true

print ""
print "+-----------------------------------------+"
print "Division:"
```

```
print 8 / 2;         // expect: 4
print 12.34 / 12.34;  // expect: 1

print ""
print "+-----------------------------------------+"
print "Equality:"

print nil == nil; // expect: true

print ""

print true == true; // expect: true
print true == false; // expect: false

print ""

print 1 == 1; // expect: true
print 1 == 2; // expect: false

print ""

print "str" == "str"; // expect: true
print "str" == "ing"; // expect: false

print ""

print nil == false; // expect: false
print false == 0; // expect: false
print 0 == "0"; // expect: false

print ""
print "+-----------------------------------------+"
print "Multiply:"

print 5 * 3; // expect: 15
print 12.34 * 0.3; // expect: 3.702

print ""
print "+-----------------------------------------+"
print "Not equals:"

print nil != nil; // expect: false

print ""

print true != true; // expect: false
print true != false; // expect: true

print ""
```

```
print 1 != 1; // expect: false
print 1 != 2; // expect: true

print ""

print "str" != "str"; // expect: false
print "str" != "ing"; // expect: true

print ""

print nil != false; // expect: true
print false != 0; // expect: true
print 0 != "0"; // expect: true


print ""
print "+-----------------------------------------+"
print "Negate:"

print !true;     // expect: false
print !false;    // expect: true
print !!true;    // expect: true
print !123;      // expect: false
print !0;        // expect: false
print !nil;     // expect: true
print !"";       // expect: false
fun foo() {}
print !foo;      // expect: false

print ""

print -(3); // expect: -3
print --(3); // expect: 3
print ---(3); // expect: -3

print ""
print "+-----------------------------------------+"
print "Returns:"

fun f() {
  if (false) "no"; else return "ok";
}

print f(); // expect: ok

print ""

fun f() {
  if (true) return "ok";
}
```

```
print f(); // expect: ok

print ""

fun f() {
  while (true) return "ok";
}

print f(); // expect: ok

print ""

print ""
print "+-------------------------------------------+"
print "Multiline printing:"

var a = "1
2
3";
print a;
// expect: 1
// expect: 2
// expect: 3
print ""
print "+-------------------------------------------+"
print "While:"
var f1;
var f2;
var f3;
var i = 1;
while (i < 4) {
  var j = i;
  fun f() { print j; }
  if (j == 1) f1 = f;
  else if (j == 2) f2 = f;
  else f3 = f;

  i = i + 1;
}
f1(); // expect: 1
f2(); // expect: 2
f3(); // expect: 3
```