```
print "Megan Mulcahy's Test suite for BYTE 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("Empty Blocks:");

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

print("+----------------------------------------+");
print("Inner & Outer Blocks:");

var a = "outer";
{
  var a = "inner";
```

```
  print(a);          // expect: inner
}
print(a);         // expect: outer

print("+-----------------------------------------+");
print("Equality");

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

print("");
// Not equal to other types.
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("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("Expressions:");

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

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("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 "" ;


print clock; // expect: <native fn>
```