Testing Your Interpreter, Part 3

Test 1: A main with code inside. This code should return 10.

```
function main() {
  var x = 10;
  var y = 20;
  var z = 30;
  var min = 0;

  if (x < y)
    min = x;
  else
    min = y;
  if (min > z)
    min = z;
  return min;
}
```

Test 2: A function that uses global variables. This code should return 14.

```
var x = 4;
var y = 6 + x;

function main() {
  return x + y;
}
```


Test 3: A function that changes global variables. This code should return 45.


```
var x = 1;
var y = 10;
var r = 0;

function main() {
  while (x < y) {
     r = r + x;
     x = x + 1;
  }
  return r;
}
```


Test 4: A recursive function. This code should return 55.

```
function fib(a) {
  if (a == 0)
    return 0;
  else if (a == 1)
    return 1;
  else
    return fib(a-1) + fib(a-2);
}

function main() {
  return fib(10);
}
```

Test 5: Functions with multiple parameters that hide global variables. This code should return 1.

```
function min(x, y, z) {
  if (x < y) {
    if (x < z)
      return x;
```

```
    else if (z < x)
      return z;
  }
  else if (y > z)
    return z;
  else
    return y;
}

var x = 10;
var y = 20;
var z = 30;

var min1 = min(x,y,z);
var min2 = min(z,y,x);

function main() {
  var min3 = min(y,z,x);

  if (min1 == min3)
    if (min1 == min2)
      if (min2 == min3)
        return 1;
  return 0;
}
```

Test 6: Verifying that your code uses static scoping instead of dynamic scoping. This code should return 115.


```
var a = 10;
var b = 20;

function bmethod() {
  var b = 30;
  return a + b;
}

function cmethod() {
  var a = 40;
  return bmethod() + a + b;
}

function main () {
  var b = 5;
  return cmethod() + a + b;
}
```

Test 7: Boolean parameters and return values. This code should return true.

```
function minmax(a, b, min) {
  if (min && a < b || !min && a > b)
    return true;
  else
    return false;
}

function main() {
  return (minmax(10, 100, true) && minmax(5, 3, false));
}
```

Test 8: Multiple function calls in an expression. This code should return 20.

```
function fact(n) {
  var f = 1;
  while (n > 1) {
    f = f * n;
```

```
    n = n - 1;
  }
  return f;
}

function binom(a, b) {
  var val = fact(a) / (fact(b) * fact(a-b));
  return val;
}

function main() {
  return binom(6,3);
}
```

Test 9: A function call in the parameter of a function. This code should return 24.

```
function fact(n) {
  var r = 1;
  while (n > 1) {
    r = r * n;
    n = n - 1;
  }
  return r;
}

function main() {
  return fact(fact(3) - fact(2));
}
```


Test 10: A function call that ignores the return value. This code should return 2.


```
var count = 0;

function f(a,b) {
  count = count + 1;
  a = a + b;
  return a;
}
```



```
function main() {
  f(1, 2);
  f(3, 4);
  return count;
}
```

Test 11: A function without a return statement. This code should return 35.

```
var x = 0;
var y = 0;

function setx(a) {
  x = a;
}

function sety(b) {
  y = b;
}

function main() {
  setx(5);
  sety(7);
  return x * y;
}
```

Test 12: Mismatched parameters and arguments. This code should give an error.

```
function f(a) {
  return a*a;
}

function main() {
  return f(10, 11, 12);
}
```

Test 13: Functions inside functions. This code should return 90.

```
function main() {
  function h() {
    return 10;
  }

  function g() {
    return 100;
  }

  return g() - h();
}
```

Test 14: Functions inside functions accessing variables outside. This code should return 69.

```
function collatz(n) {
  var counteven = 0;
  var countodd = 0;

  function evenstep(n) {
    counteven = counteven + 1;
    return n / 2;
  }

  function oddstep(n) {
    countodd = countodd + 1;
    return 3 * n + 1;
  }

  while (n != 1) {
    if (n % 2 == 0)
      n = evenstep(n);
    else
      n = oddstep(n);
  }
  return counteven + countodd;
}


function main() {
  return collatz(111);
}
```


Test 15: Functions inside functions with variables of the same name. Thus code should return 87.

```
function f(n) {
  var a;
  var b;
  var c;

  a = 2 * n;
  b = n - 10;

  function g(x) {
    var a;
    a = x + 1;
    b = 100;
    return a;
```

```
  }

  if (b == 0)
    c = g(a);
  else
    c = a / b;
  return a + b + c;
}

function main() {
  var x = f(10);
  var y = f(20);

  return x - y;
}
```

Test 16: Functions inside functions inside functions. This code should return 64.

```
function main() {
  var result;
  var base;

  function getpow(a) {
     var x;

     function setanswer(n) {
        result = n;
     }

     function recurse(m) {
       if (m > 0) {
         x = x * base;
         recurse(m-1);
       }
       else
         setanswer(x);
     }

     x = 1;
     recurse(a);
  }
  base = 2;
  getpow(6);
  return result;
}
```


Test 17: Functions inside functions accessing out of scope variables. This code should return an error with b out of scope.

```
function f(x) {
  function g(x) {
    var b;
    b = x;
    return 0;
  }

  function h(x) {
    b = x;
    return 1;
  }

  return g(x) + h(x);
}

function main() {
  return f(10);
}
```

Test 18: try/catch finally, but no exception thrown. This code should return 125.

```
function divide(x, y) {
  if (y == 0)
    throw y;
  return x / y;
}

function main() {
  var x;

  try {
    x = divide(10, 5) * 10;
    x = x + divide(5, 1);
  }
  catch(e) {
    x = e;
  }
  finally {
    x = x + 100;
  }
  return x;
}
```

Test 19: Throwing an exception inside a function. This code should return 100.

```
function divide(x, y) {
  if (y == 0)
    throw y;
  return x / y;
}

function main() {
  var x;

  try {
    x = divide(10, 5) * 10;
    x = x + divide(5, 0);
  }
  catch(e) {
    x = e;
  }
  finally {
    x = x + 100;
  }
  return x;
}
```


Test 20: Throwing an exception from a function. This code should return 2000400.

```
function divide(x, y) {
  if (y == 0)
    throw 1000000;
  return x / y;
}

 function main() {
  var x = 0;
  var j = 1;

  try {
    while (j >= 0) {
    var i = 10;
    while (i >= 0) {
      try {
        x = x + divide(10*i, i);
      }
```

```
      catch(e) {
        x = x + divide(e, j);
      }
      i = i - 1;
    }
    j = j - 1;
   }
  }
  catch (e2) {
    x = x * 2;
  }
  return x;
}
```

Additional Tests For Those Doing the Extra Challenge

Test 21: Call-by-reference test. This code should return 3421.

```
function swap1(x, y) {
  var temp = x;
  x = y;
  y = temp;
}

function swap2(&x, &y) {
  var temp = x;
  x = y;
  y = temp;
}

function main() {
  var a = 1;
  var b = 2;
  swap1(a,b);
  var c = 3;
  var d = 4;
  swap2(c,d);
  return a + 10*b + 100*c + 1000*d;
}
```


Test 22: Assignment side effects with function calls. This code should return 20332.

```
var x;

function f(a,b) {
  return a * 100 + b;
}

function fib(f) {
  var last = 0;
  var last1 = 1;

  while (f > 0) {
    f = f - 1;
    var temp = last1 + last;
    last = last1;
    last1 = temp;
  }
  return last;
}

function main() {
  var y;
  var z = f(x = fib(3), y = fib(4));
  return z * 100 + y * 10 + x;
}
```

Test 23: Mixture of call-by-value and call-by-reference. This code should return 21.

```
function gcd(a, &b) {
  if (a < b) {
    var temp = a;
    a = b;
    b = temp;
  }
  var r = a % b;
  while (r != 0) {
    a = b;
    b = r;
    r = a % b;
  }
  return b;
}
function main () {
  var x = 14;
  var y = 3 * x - 7;
  gcd(x,y);
  return x+y;
}
```