**Exercise 2  
Callan Moore**

**Exercise 2.1**

* AppleScript
* Haskell
* PL/SQL
* TSQL

**Exercise 2.2**

* \_end
* End
* NULL

**Exercise 2.3**

Stack overflow error occurs when negative number is input. Modify input to only accept n values greater or equal to zero so negative number cannot be computed.

-- defines a factorial function

function fact(n)

if n < 0 then

return nil

elseif n == 0 then

return 1

else

return n \* fact(n-1)

end

end

print("enter a number:")

a = io.read("\*n") -- reads a number

print(fact(a))

**Exercise 2.4**

Considering –l can only be used in the command line, I prefer the dofile method as it can be used in both.

**Exercise 2.5**

print(arg[0])

**Exercise 2.6**

False

Type returns a string and so when compared to the value of nil the are not equal and therefore returns false.

**Exercise 2.7**

* .0e12 0
* 0x12 18
* 0xFFFFFFF 268435455
* FFFF nil
* 0xA 10
* 0.1e1 1

**Exercise 2.8**

All the a’s are the same and all are pointing to the same memory address. Trying to make the a.a.a.a = 3 fails as 3 is not a valid memory address which is what value of a is now. However if the assignment a.a.a.a = 3 is done immediately then a.a no longer is set to the table but to a value itself.

**Exercise 2.9**

-10 2

-9 0

-8 1

-7 2

-6 0

-5 1

-4 2

-3 0

-2 1

-1 2

0 0

1 1

2 2

3 0

4 1

5 2

6 0

7 1

8 2

9 0

10 1

% is a modulus -> remainder operator.

**Exercise 2.10**

2^3^4 = 2.4178516392293e+024 Large Number  
2^-3^4 = 4.1359030627651e-025 Small Number

**Exercise 2.11**

function CalcPolynomial(poly, xValue)

result = 0

for i = 1, #poly do

result = result + poly[i] \* (xValue^(i-1))

end

print(result)

end

**Exercise 2.12**

function CalcPolynomial2(poly, xValue)

result = 0

for i = 1, #poly do

xCalc = 1

for j = 1, (i - 1) do

xCalc = xCalc \* xValue

end

result = result + poly[i] \* xCalc

end

print(result)

end

**Exercise 2.13**

if( (bCheck == true) or (bCheck == false)) then

print("Boolean")

else

print("Not a Boolean")

end

**Exercise 2.14**

No the parenthesis are not necessary as the precedence order negates the need. However YES I would use them as it makes it easier for a human to read and understand the order of logic at a glance.

**Exercise 2.15**

Monday Sunday Sunday

Its convoluted and silly, however the printing of the table results in the first the value stored in Sunday and both the other two filter down to both holding the value stored in Monday. The values of both of those are the other day.

I actually understand this, but cannot think of a concise way to put it in writing.

**Exercise 2.16**

Function AddToTable(sequence, meaning)

Table[#Table + 1] = sequence

Table[sequence] = meaning

end