Exercise 2

2.1) [Ada](https://en.wikipedia.org/wiki/Ada_(programming_language)), [Eiffel](https://en.wikipedia.org/wiki/Eiffel_(programming_language)), [Euphoria](https://en.wikipedia.org/wiki/Euphoria_(programming_language)), [Occam](https://en.wikipedia.org/wiki/Occam_(programming_language)), [SPARK](https://en.wikipedia.org/wiki/SPARK_(programming_language)), ANSI[SQL](https://en.wikipedia.org/wiki/SQL), [ToolBook OpenScript](https://en.wikipedia.org/w/index.php?title=ToolBook_OpenScript&action=edit&redlink=1), and [VHDL](https://en.wikipedia.org/wiki/VHDL)

2.2)

* \_end
* End
* NULL

2.3.1) you receive the following error.

Code Snippet:

lua: Lecture2\_Test.lua:70: stack overflow

stack traceback:

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

...

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:70: in function 'fact'

Lecture2\_Test.lua:75: in main chunk

[C]: ?

>Exit code: 1

2.3.2) Fixing it by only accepting positive numbers

Or returning nil in the event of receiving a negative number

Code Snippet:

-- 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))

2.4) Prefer dofile(“filename”) as the lua –l is only valid in command line, but dofile is valid in both.

2.5) Print the current executing script file: print(arg[0]);

2.6) False, because type(nil) return the string “nil” which is different from the value nil

2.7)

* .0e12 = 0
* 0.e12 = failed
* 0.0e = failed
* 0x12 = 18
* 0xABFG = failed
* 0xFFFFFFF = 2684435455
* FFFF = nil
* 0xA = 10
* 0x = failed
* 0x1P10 = failed
* 0.1e1 = 1
* 0x0.1p1 = failed

2.8) a.a.a.a is the same as a.a.a and a.a they all return the memory address of a. Though when you write a.a.a.a = 3, you are assigning the value 3 to a essentially therefor a.a.a.a no longer a valid line as a is no longer a table but the value 3 instead

2.9)

Output:  
-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 returns the remainder of division

2.10)

Output:  
 2^3^4: 2.4178516392293e+024 (Large positive number)  
2^-3^4: 4.1359030627651e-025 (Small negative number)

Exercise 2.11

-- Polynomial function

function Polynomial(p,x)

local result = 0;

for i = 1, #p do

result = result + p[i] \* (x^(i-1));

end

return result;

end

Exercise 2.12

-- Polynomial function

function Polynomial2(p,x)

local result = 0;

for i = 1, #p do

local expo = 1;

for j = 1, (i-1) do

expo = expo \* x;

end

result = result + p[i] \* expo;

end

return result;

end

Exercise 2.13

x = nil -- (some value)

-- Is x a boolean   
print (((x == false) or (x == true)));

Exercise 2.14

No, yes

Exercise 2.15

Sunday = "Monday";   
Monday = "Sunday";  
t = {Sunday = "Monday", [Sunday] = Monday}  
print (t.Sunday, t[Sunday], t[t.Sunday])

Output => Monday Sunday Sunday

Variable Sunday = String Value “Monday”  
Variable Monday = String Value “Sunday”

The Table t has:  
key Sunday which equals string “Monday”  
and key [Sunday] (Which take the value of variable Sunday (Which is string “Monday”) and set that as the Key) which equals the Variable Monday which equals string “Sunday”

Which means t can be written as:

t = {Sunday = “Monday”, Monday = “Sunday”}  
 or  
 t = {Sunday = Sunday, Monday = Monday};

* Then t.Sunday (same as t[“Sunday”]) is looking for key Sunday inside the table and returns its value the string “Monday”
* Then t[Sunday] (same as t[“Monday”](Outside the table)) is looking for key Monday(outside) as Sunday equates to the string “Monday”. The key Monday is not explicitly in the table but key [Sunday] equate to the key “Monday” therefor returns the value of variable Monday which is String “Sunday”
* Then t[t.Sunday](Same as t[“Monday”(Inside the table)]) is looking for the key Monday(Inside) as Sunday equates to the string “Monday”. The key Monday is not explicitly in the table but key [Sunday] equate to the key “Monday” therefor returns the value of variable Monday which is String “Sunday”

Exercise 2.16

EscapeTable = {(["/b"] = "BackSlash"), (["/n"] = "NewLine"), (["/t"] = "Tab")}