```lua
-- Week 1, Activity 18: Extending our Arithmetic Expressions
-- We'll extend our arithmetic expression processor to include
-- other operations.

local lpeg = require "lpeg"
local pt = require "pt"

vazio = -lpeg.P(1)
espaco = lpeg.S(" \n\t")^0
sinal = lpeg.S("+-")^-1
numero = ((sinal * lpeg.R("09")^1) / tonumber) * espaco
operador = lpeg.C(lpeg.S("+-")) * espaco

soma = espaco * numero * (operador * numero)^0 * vazio

print(soma:match("12 + 43 + -43 + 3"))

-- Traverse the list 2 by 2 elements, and check to
-- verify if we are adding ou subtracting:
function fold(lst)
  local acc = lst[1]
  for i = 2, #lst, 2 do
    if lst[i] == "+" then
      acc = acc + lst[i + 1]
    else
      acc = acc - lst[i + 1]
    end
  end
  return acc
end

local tabela = espaco * lpeg.Ct(numero * (operador * numero)^0) * vazio
local soma = espaco * lpeg.Ct(numero * (operador * numero)^0) / fold * vazio
local teste = "32 - 32 + 14 - 1"
print(teste)
print(pt.pt(tabela:match(teste)))
print(soma:match(teste))


-- Now we are going to extend with multiplicative operadors. We could be
-- tempted to put the multiplicative operatons inside our operator list,
-- and adjust the fold function:
vazio = -lpeg.P(1)
espaco = lpeg.S(" \n\t")^0
sinal = lpeg.S("+-")^-1
numero = ((sinal * lpeg.R("09")^1) / tonumber) * espaco
operador = lpeg.C(lpeg.S("+-*/")) * espaco

function fold(lst)
  local acc = lst[1]
  for i = 2, #lst, 2 do
    if lst[i] == "+" then
      acc = acc + lst[i + 1]
    elseif lst[i] == "-" then
      acc = acc - lst[i + 1]
    elseif lst[i] == "*" then
      acc = acc * lst[i + 1]
    elseif lst[i] == "/" then
      acc = acc / lst[i + 1]
    else
```

```
      error("unknown operator")
    end
  end
  return acc
end

-- But this does not consider the precedende of operators. The result, in
-- this example, should be 5 but we get 6:
local tabela = espaco * lpeg.Ct(numero * (operador * numero)^0) * vazio
local soma = espaco * lpeg.Ct(numero * (operador * numero)^0) / fold * vazio
local teste = "1 + 3 * 2 - 2"
print(teste)
print(pt.pt(tabela:match(teste)))
print(soma:match(teste))

-- The correct way to deal with this problem is to have 2 sets of operators:
-- the additive and the multiplicative operators, like this:
vazio = -lpeg.P(1)
espaco = lpeg.S(" \n\t")^0
sinal = lpeg.S("+-")^-1
numero = ((sinal * lpeg.R("09")^1) / tonumber) * espaco
opAdd = lpeg.C(lpeg.S("+-")) * espaco
opMul = lpeg.C(lpeg.S("*/")) * espaco

-- And to have 2 sets of captures: one for multiplicative operators (let's
-- call it a term) and other for additive operators (let's call it a sum:
local term = espaco * lpeg.Ct(numero * (opMul * numero)^0) / fold
local sum = espaco * lpeg.Ct(term * (opAdd * term)^0) / fold * vazio
-- The idea here is to treat everything as a term, but the multiplicative
-- operators are treated first (we are summing terms, but a term could be
-- a multiplicative operation: the sum is made after the multiplicative
-- terms were calculated).

-- The fold function does not need any modification:
function fold(lst)
  local acc = lst[1]
  for i = 2, #lst, 2 do
    if lst[i] == "+" then
      acc = acc + lst[i + 1]
    elseif lst[i] == "-" then
      acc = acc - lst[i + 1]
    elseif lst[i] == "*" then
      acc = acc * lst[i + 1]
    elseif lst[i] == "/" then
      acc = acc / lst[i + 1]
    else
      error("unknown operator")
    end
  end
  return acc
end

-- Let's see:
local teste = "1 + 3 * 2 - 2"
local tabela = espaco * lpeg.Ct(numero * (operador * numero)^0) * vazio
print(teste)
print(pt.pt(tabela:match(teste)))
print(sum:match(teste))

local teste = "2+14*2/2-30"
```

```
local tabela = espaco * lpeg.Ct(numero * (operador * numero)^0) * vazio
print(teste)
print(pt.pt(tabela:match(teste)))
print(sum:match(teste))
```