```lua
-- Week1, Activity 10: Concatenation, Repetition and Choices
-- How to build more complex patterns from simple ones

local lpeg = require "lpeg"


-- Concatenation of patterns: *
p = lpeg.P("a") * lpeg.P("h4")
print(p:match("ah4"))              --> 4

-- When using operators in LPeg, if one of the operands
-- are not a pattern, LPeg convert the operand in a pattern.
-- (at least one of the operands must be a pattern)
q = lpeg.P("a") * "rara"
print(q:match("arara"))            --> 6

-- What this pattern do?
r = lpeg.P(3) * lpeg.P("oi")

-- As lpeg.P(3) matches exaclty 3 characters, whatever
-- they are, this pattern matches any 3 characters
-- followed by "oi". Ex.:
print(r:match("   oi"))            --> 6
print(r:match("z4!oi"))            --> 6
print(r:match("12oi"))             --> nil



-- Repetition in LPeg is done with ^ operator.
-- ^n    matches n or more times
-- ^-n   matches at most n times (optional)
a = lpeg.P("a")^0
print(a:match("aaaa"))             --> 5
print(a:match(""))                 --> 1
print(a:match("aaaaaaaaaaaaa"))    --> 14
b = lpeg.P("a")^1
c = lpeg.P("a")^3

-- Repetitions can also be specified with negative
-- powers (at most N times). It indicates something
-- that is optional!
d = lpeg.P("a")^-2
print(d:match("aa"))               --> 3
print(d:match("baa"))              --> 1
print(d:match("a"))                --> 2

-- What is this pattern?
e = lpeg.P(1)^-1
-- one single optional character!



-- We can also use the + operator, meaning an OR
-- of patterns:
f = lpeg.P("one") + lpeg.P("two")
print(f:match("one"))              --> 4
print(f:match("two"))              --> 4
g = lpeg.S("aeiou") + lpeg.S("67")
print(g:match("e"))                --> 2
print(g:match("7"))                --> 2
```

```lua
-- IMPORTANT! LPeg is POSSESSIVE! That means that
-- if LPeg matches the first operand, it will never
-- try for the other operands (even if that fails
-- later).
h = lpeg.P("a") + lpeg.P("abc")
print(h:match("abc"))              --> 2

i = (lpeg.P("a") + lpeg.P("abc")) * "d"
print(i:match("ad"))               --> 3
print(i:match("abcd"))             --> nil (possessive behavior)

j = lpeg.P("a")^0 * lpeg.P("a")
print(j:match("aaa"))              --> nil (possessie behavior)

-- Possessiveness general rule:
-- Choices (+) and Repetitions (^) are always possessive!
--
-- Choices (+):     always matches the first option if succeded
-- Repetitions (^): always matches the maximum possible
--
-- This possessive behavior is a feature that permits LPeg
-- deterministic pattern matching! It's not much flexible,
-- but it allowed us total control over the matching.
```