Funkcionális programozás 7. gyakorlat

1. Definiáld a faktoriális függvényt rekurzívan!

```
fact 0 == 1
fact 1 == 1
fact 3 == 6
fact 6 == 720
```

2. Számold ki az n. Fibonacci-számot!

```
fib 0 == 0
fib 1 == 1
fib 2 == 1
fib 4 == 3
fib 5 == 5
```

3. *A `fib` függvény eléggé lassú még kis számokra is. Ennek oka, hogy `fib` újra és újra kiszámol bizonyos Fibonacci-számokat. Tudsz találni egy gyorsabb megoldást?

4. Definiáld a hatvány függvényt! Haskellben ez a `^` operátor, nevezzük el a sajátunkat `pow`-nak. Definiáld rekurzívan, ne használd a `^`-t!

```
pow 0 2 == 0
pow 2 0 == 1
pow 2 1 == 2
pow 3 2 == 9
```

5. Definiálj egy `range` függvényt, mely felsorolja az egész számokat egy alsó és egy felső korlát között! Ne használd a `[ .. ]` kifejezést! Feltesszük, hogy a második paraméter nem kisebb, mint az első.

```
range 5 9 == [5, 6, 7, 8, 9]
range 5 5 == [5]
range 0 3 == [0, 1, 2, 3]
```

6. Módosítsd a `range` függvényt, hogy csökkenő sorozatot is elő tudjon állítani, ha a második paraméter kisebb, mint az első!

```
range 5 9 == [5, 6, 7, 8, 9]
range 5 5 == [5]
range 0 3 == [0, 1, 2, 3]
```

7. Definiáld újra a `length` függvényt, mely megszámolja egy lista elemeit!

```
length' []      == 0
length' [5]     == 1
length' [8,0,3] == 3
```

8. Definiáld újra a minimum függvényt, mely megkeresi egy lista legkisebb elemét!

```
minimum' [0] == 0
minimum' [9, 3, 4, 1, 10] == 1
```

9. Definiálj egy függvényt, mely kigyűjti egy lista minden második elemét!

```
everySecond "Haskell" == "akl"
everySecond "H"       == ""
everySecond "java"    == "aa"
everySecond ""        == ""
```

10. Definiáld újra az `elem` függvényt, mely rekurzívan leellenőrzi, hogy egy elem megtalálható-e egy listában!

```
elem' 'l' "Haskell"
not (elem' 'v' "Lujzi")
not (elem' 'x' "")
```

11. Definiálj egy függvényt, mely kikeresi egy kulcs-érték párokból álló listában egy kulcshoz tartozó értéket! Ha a kulcs nem található, adj hibaüzenetet az `error` függvénnyel!

```
value 5 [(0,"c++"),(5,"python"),(4,"rust")] == "python"
value 4 [(0,"c++"),(5,"python"),(4,"rust")] == "rust"
value 4 [(0,"c++"),(5,"python"),(4,"go")]   == "go"
```

12. Módosítsd a `value` függvényt, hogy egy alapértelmezett értéket adjon vissza, ha a keresett kulcs nem található a listában!

```
value 5 "scala" [(0, "c++"), (5, "python")] == "python"
value 3 "scala" [(0, "c++"), (1, "java"), (5, "python"), (4, "rust")] == "scala"
```