Funkcionális programozás 4. gyakorlat

Rekurzió

http://lambda.inf.elte.hu/Recursion.xml

- Egy függvény közvetlenül, vagy közvetve önmagát hívja
- Pl.:

```
fact 0 = 1
fact n = n * fact (n-1)
```

Kiértékelődés:

```
fact 3
3 * (fact 2)
3 * (2 * (fact 1))
3 * (2 * (1 * (fact 0)))
3 * (2 * (1 * (1)))
3 * (2 * 1)
3 * 2
6
```

Összegzés

Készítsünk egy függvényt, amely vár egy számot és 0..n-ig összeadja a számokat. (Először tegyük fel, hogy csak pozitív számot adhatunk meg. Később próbáljuk meg, hogy ha negatív számot kapunk, akkor hogyan alakítsuk a függvényt!)

```
sumN 0 == 0
sumN 3 == 6
sumN 8 == 36
```

Fibonacci

Definiálj függvényt, amely visszaadja az n. Fibonacci-számot!

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

Minták listákra

http://lambda.inf.elte.hu/Patterns.xml#mint%C3%A1k-list%C3%A1kra

- Üres lista minta: `[]`
- Egyelemű lista: `[a]`
- Kételemű lista: `[a,b]`

- Nemüres lista minta: `a:b`

Fejelem (head) lekérése: `head [1,2,3,4] == 1`

Farok rész (tail) lekérése: `tail [1,2,3,4] == [2,3,4]`

Utolsó elem lekérése: `last [1,2,3,4] == 4`

Lista lekérése az utolsó elem nélkül: `init [1,2,3,4] == [1,2,3]`

Lista ürességének ellenőrzése: `null [] == True`

Elem hozzáfűzése egy lista elejére

```
5 : [1,2,3] == [5,1,2,3]
1 : 2 : 3 : [] == [1,2,3]
```

Definiáljuk a null' függvényt mintaillesztés segítségével, amely eldönti egy listáról, hogy üres-e!

```
null' :: [a] -> Bool
```

Definiáljuk a head' függvényt mintaillesztés segítségével, amely visszaadja egy lista első elemét!

```
head' :: [a] -> a
```

Definiáljuk az "1 elemű-e a lista" függvényt mintaillesztéssel!

```
isSingleton :: [a] -> Bool
```

Definiáljunk egy olyan függvényt, amely eldönti, hogy 0-val kezdődik-e a lista!

```
headZero :: (Num a, Eq a) => [a] -> Bool
```

Definiáljuk újra a tail függvényt, amely megadja egy lista fejelem utáni részét!

```
tail' :: [a] -> [a]
```

Definiáljuk újra a length függvényt, amely megadja egy lista hosszát!

```
length' :: [a] -> Int
```

Definiáljuk újra a sum függvényt, amely összegez egy számlistát!

```
sum' :: Num a => [a] -> a
```

Definiáljuk újra a last függvényt, amely visszaadja egy lista utolsó elemét!

```
last' :: [a]{-nemüres-} -> a
```

Definiáljuk újra az init függvényt, amely egy lista összes elemét visszaadja, az utolsót kivéve!

```
init' :: [a]{-nemüres-} -> [a]
```