Funkcionális programozás 5. gyakorlat

1. Definiáld újra a `head` függvényt egész számok listáján!

   ```
   headInt [5, 6, 7] == 5
   headInt [10]      == 10
   ```

2. Definiáld újra a `tail` függvényt egész számok listáján!

   ```
   tailInt [5, 6, 7] == [6, 7]
   tailInt [10]      == []
   ```

3. Definiáld újra a `null` függvényt egész számok listáján!

   ```
   nullInt []
   not (nullInt [1, 2, 3])
   not (nullInt [1..])
   ```

4. Definiáld az `isSingletonInt` függvényt, mely megállapítja, hogy egészek egy listája pontosan egy elemet tartalmaz-e!

   ```
   not (isSingletonInt [])
   isSingletonInt [5]
   not (isSingletonInt [6, 8])
   not (isSingletonInt [5..])
   ```

5. Definiálj egy `toUpperFirst` függvényt, mely egy szöveg első betűjét nagybetűre cseréli! Üres szöveget változatlanul adja vissza a függvény!

   ```
   toUpperFirst ""                == ""
   toUpperFirst "finn the human" == "Finn the human"
   toUpperFirst "jake"            == "Jake"
   ```

6. Definiáld újra az `isLetter` függvényt! Elég csak az angol ábécé betűit felismerni.

   ```
   isLetter 'a'
   isLetter 'A'
   isLetter 'b'
   isLetter 'X'
   not (isLetter '?')
   ```

   Tipp: itt jól jöhet az `elem` függvény, mely megnézi, hogy egy elem megtalálható-e egy listában:

   ```
   elem 5 [1,2,3,4,5]
   not (elem 0 [1,2,3,4,5])
   ```

7. Definiáld újra az `isDigit` függvényt! Elég csak a tízes számrendszer számjegyeit felismerni.

   ```
   isDigit '0'
   isDigit '5'
   isDigit '9'
   not (isDigit 'a')
   not (isDigit ' ')
   ```

8. Definiálj egy függvényt, mely egy növekvő majd csökkenő számlistát állít elő!

   ```
   mountain 3 == [1, 2, 3, 2, 1]
   mountain 5 == [1, 2, 3, 4, 5, 4, 3, 2, 1]
   ```

9. Definiálj egy függvényt, mely előállítja egy szám osztóit! 0 osztója az összes természetes szám.

   ```
   divisors 10 == [1, 2, 5, 10]
   divisors 16 == [1, 2, 4, 8, 16]
   divisors 3  == [1, 3]
   ```

10. Definiálj egy konstanst, mely kettő hatványainak végtelen listáját tárolja!

    ```
    take 5 powersOfTwo  == [1, 2, 4, 8, 16]
    take 10 powersOfTwo ==
      [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
    ```

11. *Definiálj egy konstanst, amely megadja π közelítő értékét a Leibniz-féle sor részleges előállításával! A képlet a következő:

    $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \dots$

    Természetesen ezt mi csak közelítjük, így elég az első ezer elem összegét venni, majd az eredményt néggyel megszorozni.

    Segítség: Először érdemes az `[1, -3, 5, -7, 9, -11, ... ]` végtelen listát előállítani. Majd vedd ezen elemek reciprokát és add őket össze!

    Megjegyzés: `pi` már létezik Haskellben.

12. Állíts elő egy listát, amely sorrendben tartalmazza az összes óra-perc párt!

    ```
    length time == 1440
    ```