Deklaratív programozás, 1. gyakorlat (Elixir 1), 2021.09.14. -- 2021.09.16.

Az alábbi feladatok megoldására írjon olyan Elixir-függvényeket, amelyek megfelelnek a fejkommenteknek. Írjon mindegyikből több verziót.

A függvények első verziója rekurzív legyen, a Kernel modulon kívül más modulok függvényeit ne használják, listajelölőket se használjanak. A további verziókban már ezeket is lehet használni. A feladatok megoldásához használhatja a korábbi feladatokban definiált függvényeit.

A függvényeket az IEx-ben a modulnév megadásával kell meghívni (Mnév.fnév). Az alábbi példákban nem írjuk ki a modulnevet.

---

1. Legnagyobb közös osztó euklideszi algoritmussal

```
   @spec lnko(a :: integer, b :: integer) :: d :: integer
   # a és b legnagyobb közös osztója d

   lnko(96, 42) === 6
```

---

2. Lista hossza akkumulátor nélkül és akkumulátort használó segédfüggvénnyel

```
   @spec len(xs :: [any])  :: n :: integer
   # Az xs lista hossza n

   len([]) === 0
   len('hosszú_karakterlista') === 20

   @spec leni(xs :: [any]) :: n :: integer
   # Az xs lista hossza n
   @spec leni(xs :: [any], acc :: integer) :: n :: integer
   # az xs lista hossza + acc = n

   leni([]) === 0
   leni('hosszú_karakterlista') === 20
```

---

3. Pi (a Ludolph-féle szám) közelítése a Leibniz-féle sorral
   (pi/4 = 1 - 1/3 + 1/5 - 1/7 + ...)

```
   @spec pi(i :: integer) :: pi :: float
   # A pi i-edik közelítő értéke pi

   abs(pi(10000000) - :math.pi) < 1.0e-6
```

---

4. Egész szám tetszőleges számrendszerbe konvertálása

```
   @spec dec2rad(r :: integer, i :: integer) :: ds :: [integer]
   # Az i egész szám r alapú számrendszerbe konvertált, decimális számként
   # megadott számjegyeinek listája ds

   dec2rad(2, 13)   === [1, 1, 0, 1]
   dec2rad(17, 127) === [7, 8]
```

---

5. Lista utolsó eleme Erlang-, ill. Elixir-stílusú hibakezeléssel
   Mutassa be, hogyan használhatók mintaillesztéssel a függvények

```
   @spec last_er(xs::[any]) :: r :: ({:ok, x::any} | :error)
   # Ha xs üres, r == :error, különben {:ok, x}, ahol x az xs utolsó eleme

   last_er([5, 1, 2, 8, 7]) === {:ok, 7}
   last_er([])              === :error

   @spec last_ex(xs::[any]) :: r :: (x::any | nil)
   # Ha xs üres, r == nil, különben x, ahol x az xs utolsó eleme

   last_ex([5, 1, 2, 8, 7]) === 7
   last_ex([])              === nil
```

---

6. Lista adott sorszámú eleme

```
    @spec at(es::[any], n::integer) :: r :: (e::any | nil)
    # Az es lista n-edik eleme e (indexelés 0-tól)

    at([:a,:b,:c], 2) === :c
```

---

7. Lista kettévágása

```
    @spec split(ls::[any], n::integer) :: {ps::[any], ss::[any]}
    # Az ls lista n hosszú prefixuma (első n eleme) ps, length(ls) - n
    # hosszú szuffixuma (első n eleme utáni része) pedig ss

    split([10,20,30,40,50], 3) === {[10,20,30],[40,50]}
```

---

8. Lista adott hosszúságú prefixuma

```
    @spec take(xs::[any], n::integer) :: ps::[any]
    # Az xs lista n hosszú prefixuma a ps lista

    take([10,20,30,40,50], 3) === [10,20,30]
```

---

9. Lista adott hosszúságú része utáni szuffixuma

```
    @spec drop(xs::[any], n::integer) :: ss::[any]
    # Az xs lista első n elemét nem tartalmazó szuffixuma ss

    drop([10,20,30,40,50], 3) === [40,50]
```

---

10. Lista egyre rövidülő szuffixumainak listája

```
    @spec tails(xs::[any]) :: zss::[[any]]
    # Az xs lista egyre rövidülő szuffixumainak listája zss

    tails([1,4,2]) === [[1,4,2],[4,2],[2],[]]
    tails([:a,:b,:c,:d]) === [[:a,:b,:c,:d],[:b,:c,:d],[:c,:d],[:d],[]]
```

---

11. Listában párosával előforduló elemek listája

```
    @spec parban(es::[any]) :: zs::[any]
    # Az es lista összes olyan elemének listája zs, amely
    # után vele azonos értékű elem áll

    parban([:a,:a,:a,2,3,3,:a,2,:b,:b,4,4]) === [:a,:a,3,:b,4]
```

---

12. Egy lista "értékei", azaz {:v, ertek} alakú elemei 2. tagjának listája

```
    @spec vertekek(xs::[any]) :: es::[any]
    # Az xs lista elemei közül a {v::atom, e:any} mintára illeszkedő
    # párok 2. tagjából képzett lista es

    vertekek([:alma, {:s,3}, {:v,1}, 3, {:v,2}]) === [1,2]
```

---

13. Listában párosával előforduló részlisták listája
    (javaslat: haszálja a tails/1, split/2 és take/2 függvényeket)

```
    @spec dadogo(xs::[any]) :: zss::[[any]]
    # zss az xs lista összes olyan nemüres (folytonos) részlistájából
    # álló lista, amelyet vele azonos értékű részlista követ

    dadogo([:a,:a,:a,2,3,3,:a,:b,:b,:b,:b]) === [[:a],[:a],[3],[:b],[:b,:b],[:b],[:b]]
```

------ $LastChangedDate: 2021-09-16 07:52:53 +0200 (cs, 16 szept 2021) $ --------