

Deklaratív programozás, Elixir, gyakorló feladatok 1

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

A függvények első verziója lehetőleg 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 vagy a tantermi gyakorlatok során definiált függvényeit.

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

1. Egész szám 2..16 alapú számrendszerbe konvertálása dec2rad/2 felhasználásával

```
@spec conv(r::integer, i::integer) :: res::{:ok, ds::charlist} | :error
# Az i decimális egész r alapú számrendszerbe konvertált jegyeinek listája ds
# res = {:ok, ds}, ha 2 <= r <= 16, egyébként :error

conv(16, 127) === {:ok, '7f'}
conv(17, 127) === :error
```

2. Lista egyre hosszabbodó prefixumainak listája

```
@spec prefixes(xs::[any]) :: zss::[[any]]
# Az xs lista egyre hosszabbodó prefixumainak listája zss

prefixes([:a, :b, :c]) === [[], [:a], [:a, :b], [:a, :b, :c]]
```

3. Lista adott hosszúságú összes részlistáját tartalmazó lista

```
@spec sublists(xs::[any], n::integer) ::
  pss::{{b::integer, ps::[any], a::integer}}
# Az xs lista olyan (folytonos) részlistái az n hosszú ps listák
# -- a pss eredménylista elemeinek középső tagja --, amelyek előtt
# b és amelyek után a számú elem áll xs-ben

sublists([:a, :b, :c], 1) === [{0, [:a], 2}, {1, [:b], 1}, {2, [:c], 0}]
sublists([:a, :b, :c], 2) === [{0, [:a, :b], 1}, {1, [:b, :c], 0}]
```

4. Lista részlistája

- rekurzióval, más függvények felhasználása nélkül;
- a take/2 és drop/2 felhasználásával;
- listajelölővel és a Enum.at/2 felhasználásával.

```
@spec sublist(xs::[any], s::integer, n::integer) :: rs::[any]
# Az xs lista s-edik elemétől kezdődő és n hosszú részlistája rs

sublist([:a, :b, :c], 2, 1) === [:b]
sublist([:a, :b, :c], 2, 2) === [:b, :c]
```

5. Listák összefűzése

- List.foldl/3-mal;
- List.foldr/3-mal.

```
@spec append(xs::[any], ys::[any]) :: zs::[any]
# Az xs lista ys elé fűzésének eredménye zs, azaz zs === xs ++ ys

append([:a, :b, :c], [1, 2, 3]) === [:a, :b, :c, 1, 2, 3]

@spec revapp(xs::[any], ys::[any]) :: zs::[any]
# A megfordított xs lista ys elé fűzésének eredménye zs,
# azaz zs === Enum.reverse(xs) ++ ys

revapp([:a, :b, :c], [1, 2, 3]) === [:c, :b, :a, 1, 2, 3]
```

6. Lista egyre rövidülő szuffixumainak listája (tails/1 újra)

- a) List.foldr/3-mal;
- b) listajelölővel és drop/2-vel.

```
@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],[[]]]
```

7. Lista kilapítása

```
@spec flatten(dls::[any]) :: rs::[any]
# A tetszőleges mélységű beágyazott listákból álló dls (deeplist)
# lista elemeiből álló, listaelemeket már nem tartalmazó lista az rs

flatten([1, [2,a], [[[[4]]], [5,["abc"]]]) ===
=== [1,2,:a,4,5,?a,?b,?c] === [1,2,:a,4,5,97,98,99]
```

8. Lista darabokra szabdalása

- a) append/3-mal;
- b) revapp/3-mal.

```
@spec slash(xs::[any], (f::[any] -> {ps::[any], ms::[any]})) :: zss::[any]
# Az xs listából az f ismételt alkalmazásával kivágott elemek listája zss, ahol
# f eredménye a {ps, ms} pár: ps lesz a zss következő eleme, ms pedig az xs még
# feldolgozásra váró része

slash('jaaaj!!! nem jooo!', fn(xs) -> Enum.split(xs, 3) end) ===
['jaa','aj!','!! ','nem','jo','oo!']
```

9. Lista első monoton növekvő futama és maradéklistája

- a) when és magasabb rendű függvények nélkül;
- b) when-nel, de magasabb rendű függvények nélkül;
- c) magasabb rendű függvényekkel.

```
@spec rampa(xs::[any()]) :: psms::{ps::[any()], ms::[any]}
# Az xs lista első monoton növekvő futamából (részlistájából),
# ps-ből és az xs maradéklistájából, ms-ből álló pár psms

rampa('abbéli') === {'abbé', 'li'}
```

10. Lista monoton növekvő futamai

- a) rampa/1-gyel, slash/1 nélkül;
- b) rampal-gyel és slash/1-gyel.

```
@spec rampak(xs::[any()]) :: xss::[[any()]]
# Az xs lista monoton növekvő futamainak (részlistáinak) listája xss

rampak('lábbelik') === ['lá','bbel','ik']
```

----- \$LastChangedDate: 2022-09-12 22:57:50 +0200 (h, 12 szept 2022) \$ -----