Deklaratív programozás, 1. gyakorlat (Erlang 1), 2020.09.22. -- 2020.09.24.

Írjon olyan Erlang-függvényt, amely megfelel az adott fejkommentnek. A feladat megoldásához felhasználhat korábbi sorszámú feladatokban definiált eljárásokat. Néhány feladathoz segítséget talál a feladatsor végén.
A függvényt a modulnév megadásával kell meghívni (mnév:fnév); a példákban elhagyjuk.

---

1. A legnagyobb közös osztó meghatározása az euklideszi algoritmussal

```
-spec lnko(A :: integer(), B :: integer()) -> D :: integer().
%% A és B legnagyobb közös osztója D.

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

---

2. Pi (a Ludolph-féle szám) közelítő meghatározása 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.
```

---

3. 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].
```

---

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

```
-spec conv(R :: integer(), I :: integer()) -> {ok, Ds :: string()} | error.
%% Az I egész szám R alapú számrendszerbe konvertált jegyeinek listája Ds,
%% ha 2 <= R <=16, egyébként error.

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

---

5. Lista utolsó elemének meghatározása hibakezeléssel
   Mutassa be, hogyan használható a safe_last függvény (tipp a végén).

```
-spec safe_last(Xs::[any()]) -> {ok, X::any()} | error.
%% Ha Xs üres, akkor 'error', különben X az Xs lista utolsó eleme.

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

---

6. Lista adott sorszámú eleme (lists:nth/2)

```
-spec nth(N::integer(), L::[any()]) -> E::any().
%% Az L lista N-edik eleme E (1-től számozva az elemeket).

nth(3, [a,b,c]) =:= c.
```

---

7. Lista kettévágása (vö. lists:split/2)

```
-spec split(N::integer(), L::[any()]) -> {P::[any()], S::[any()]}.
%% A P lista az L lista N hosszú prefixuma (első N eleme), az S lista az
%% L lista (length(L) - N) hosszú szuffixuma (első N eleme utáni része).

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

---

8. Lista adott hosszúságú prefixuma (vö. lists:sublist/2)

```
-spec take(L0::[any()], N::integer()) -> L::[any()].
%% Az L0 lista N hosszú prefixuma az L lista.

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

---

9. Lista adott hosszúságú része utáni szuffixuma (vö. lists:nthtail/2)

```
-spec drop(L0::[any()], N::integer()) -> L::[any()].
%% Az L0 lista első N elemét nem tartalmazó szuffixuma az L lista.

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 a Zss lista.

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

---

11. Lista egyre hosszabbodó prefixumainak listája

```
-spec prefixes(Xs::[any()]) -> Zss::[[any()]].
%% Az Xs lista egyre hosszabbodó prefixumainak listája a Zss lista.

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

---

12. Lista adott hosszúságú összes részlistáját tartalmazó lista

```
-spec sublists(N::integer(), Xs::[any()]) ->
                    [{B::integer(), Ps::[any()], A::integer()}].
%% Az Xs lista egy olyan (folytonos) részlistája az N hosszú Ps lista,
%% amely előtt B és amely után A számú elem áll Xs-ben.

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

---

13. Listában párosával előforduló elemek listája

```
-spec parban(Xs::[any()]) -> Zs::[any()].
%% A Zs lista az Xs lista összes olyan elemének listája, amelyet
%% vele azonos értékű elem követ.

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

---

14. Lista részlistája
    a) rekurzióval, más függvények felhasználása nélkül;
    b) a take/2 és drop/2 felhasználásával;
    c) listanézettel, a lists:nth/2 és lists:seq/2 felhasználásával.

```
-spec sublist(L0::[any()], S::integer(), N::integer()) -> L::[any()].
%% Az L0 lista S-edik elemétől kezdődő és N hosszú részlistája az L lista.

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

---

15. Egy lista "értékei", azaz {v,Ertek} alakú elemei második tagjának listája.

```
-spec vertekek(Xs::[any()]) -> Es::[any()].
%% Az Xs lista elemei közül a {v::atom(),E:any()} mintára illeszkedő
%% párok második tagjából képzett lista az Es lista.

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

---

16. Listák összefűzése a lists:foldr/3 függvénnyel

```
-spec append(Xs::[any()], Ys::[any()]) -> Zs::[any()].
%% Az Xs lista Ys elé fűzésének eredménye a Zs lista, azaz Zs =:= Xs ++ Ys.

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

---

17. Lista megfordítása és egy másik elé fűzése a lists:foldl/3 függvénnyel

```
    -spec revapp(Xs::[any()], Ys::[any()]) -> Zs::[any()].
    %% A megfordított Xs lista Ys elé fűzésének eredménye a Zs lista,
    %% azaz Zs =:= lists:reverse(Xs)++Ys.

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

---

18. A 10. feladat (tails/1) újbóli megoldása
    a) a lists:foldr/3 függvénnyel;
    b) listanézettel, a drop/2 és lists:seq/2 függvénnyel.

---

19. A 12. feladat (sublists/2) újbóli megoldása listanézettel, a take/2, drop/2, lists:seq/2 és lenght/1 függvényekkel

---

20. A 13. feladat (parban/1) újbóli megoldása
    a) listanézettel és a tails/1 függvénnyel;
    b) listanézettel, a lists:zip/2, take/2 és length/1 függvényekkel.

---

21. Listában párosával előforduló részlisták listája listanézettel, a tails/1, split/2, take/2 és lists:seq/2 függvényekkel

```
    -spec dadogo(Xs::[any()]) -> Zss::[[any()]].
    %% A Zss lista 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]].
```

---

22. Lista kilapítása (lists:flatten/1)
    a) A lists:append/2 (++) függvénnyel;
    b) lists:append/2 nélkül, egy flatten/2 segédfüggvény bevezetésével.

```
    -spec flatten(DeepList::list()) -> L::list(). %% list() =:= [any()].
    %% A tetszőleges mélységű beágyazott listákból álló DeepList lista
    %% elemeiből álló, listákat már nem tartalmazó lista az L lista.

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

SEGÍTSÉG A MEGOLDÁSHOZ

5. Használat: pl. a 'case' szerkezettel eldönthető, hogy történt-e hiba.
11. Javasolt segédfüggvény: lista legalább N hosszú prefixumainak listája.
    Hasznos BIF: length(L), azaz az L lista hossza.
14.c), 18.b), 19. és 21. "Ciklus" szervezésére használja a lists:seq/2 függvényt.
    -spec lists:seq(N::integer(), M::integer()) -> S::[integer()].
    %% S =:= [N,N+1,...,M], pl. lists:seq(1, length("abc")) =:= [1,2,3].
16., 17. Érdemes összevetni a tanult append/2 és foldr/3, ill. revapp/2 és foldl/3 függvények kódját. Javasolt segédfüggvény: a Céklából ismert cons/2.

TOVÁBBI GYAKORLÓ FELADATOK OTTHONRA

+1. Beszúrás listába adott helyre

```
    -spec insert_nth(Ls::[any()], E::any(), N::integer()) -> Rs::[any()].
    %% Az Rs lista az Ls lista olyan másolata, amelybe az Ls lista N-edik
    %% eleme után be van szúrva az E elem (a lista számozása 1-től kezdődik).
    insert_nth([1,8,3,5], 6, 2) =:= [1,8,6,3,5].
    insert_nth([1,3,8,5], 3, 3) =:= [1,3,8,3,5].
```

+2. Beszúrás rendezett listába

```
    -spec insert_ord(S0s::[any()], E::any()) -> Ss::[any()].
    %% Az Ss szigorúan monoton növekvő egészlista az S0s szigorúan monoton
    %% növekvő egészlistának az E egésszel bővített változata, ha E nem eleme
    %% az S0s listának, egyébként Ss =:= S0s.
    insert_ord([1,3,5,8], 6) =:= [1,3,5,6,8].
    insert_ord([1,3,5,8], 3) =:= [1,3,5,8].
```

+3. Adott lista sorszámozott elemeiből álló lista (vö. lists:zip/2, lists:seq/2)

```
    -spec zipseq(Ls::any()) -> {N::integer(), E::any()}.
    %% Az Ls lista N-edik eleme E (1-től számozva az elemeket).
    zipseq([a,b,c]) =:= [{1,a},{2,b},{3,c}].
```

```
+4. Lista darabokra szabdalása
    -spec slash(F::fun((list()) -> {any(),list()}), Xs::(list())) -> Zs::(list()).
    %% A Zs lista az Xs listából az F ismételt alkalmazásával előálló lista, ahol
    %% F eredménye egy pár: Zs következő eleme és Xs még fel nem dolgozott része.
    slash(fun(Xs) -> lists:split(3, Xs) end, "jaaaj!!! nem jooo!") =:=
      ["jaa","aj!","!! ","nem"," jo","oo!"].
+5. Lista monoton növekvő részlistáinak (futamainak) listája
    -spec rampak(Xs::[any()]) -> Xss::[[any()]].
    %% Az Xss lista az Xs lista monoton növekvő futamainak listája.
    rampak([1,2,2,3,2,4,5,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1]) =:=
    [[1,2,2,3],[2,4,5,6,7],[6,8],[2,3,3,4,5,6],[0,6],[5],[4],[3],[2],[1]].
    %% Alternatív megoldás: rampak/1 slash függvénnyel.
+6. Lista számtani sorozatot alkotó prefixuma
    -spec dif(Ls::[number()]) -> {Ds::[number()], Zs::[number()]}.
    %% A Ds lista az Ls lista számtani sorozatot alkotó prefixuma, a Zs az Ls
    %% maradéka (Zs utáni része).
    %%%% Segédfüggvény gyujtőargumentummal.
    -spec dif(Ls::[number()],N::number(),Rs::[number()]) ->
    %%                                    {Ds::[number()], Zs::[number()]}.
    %% A Ds lista az Ls lista N különbségu számtani sorozatot alkotó prefixuma
    %% az Rs lista fordítottja mögé fuzve, a Zs az Ls maradéka (Zs utáni része).
    dif([1,2,3,4,8,16,32,33,34]) =:= {[1,2,3,4],[8,16,32,33,34]}.
    dif([1,2,3,4,8,16,24,32,33,34]) =:= {[1,2,3,4],[8,16,24,32,33,34]}.
    %% Alternatív megoldás: dif/1 feltétel helyett mintaillesztéssel.
+7. Lista számtani sorozatot alkotó részlistáinak listája
    -spec difek(Xs::[number()]) -> Dss::[[number()]].
    %% A Dss lista az Xs lista számtani sorozatot alkotó részlistáinak listája.
    difek([1,2,3,4,8,16,32,33,34]) =:= [[1,2,3,4],[8,16],[32,33,34]].
    difek([1,2,3,4,8,16,24,32,33,34]) =:= [[1,2,3,4],[8,16,24,32],[33,34]].
    %% Alternatív megoldás: difek/1 slash függvénnyel.
    %% Feladat: úgy módosítani dif/1-et, hogy ha egy szám az egyik
    %% számtani sorozat utolsó és egyben a következő első eleme is
    %% lehet, akkor mindkettőben vegyük figyelembe
    %% NB. Ilyenkor egy sorozat záróeleme mindenképpen egy következő
    %% sorozat kezdőeleme is lesz egyben, hiszen már két elem is
    %% számtani sorozatot alkot.
    slash(fun dif1/1,[1,2,3,4,5,6,7,8,16,24,32,33,34]) =:=
      [[1,2,3,4,5,6,7,8],[8,16,24,32],[32,33,34]].
      slash(fun dif1/1,[1,2,3,4,5,6,7,16,24,32,33,34]) =:=
        [[1,2,3,4,5,6,7],[7,16],[16,24,32],[32,33,34]].
+8. Mátrix részmátrixa, valósítsa meg többféleképpen is
    -spec kozepe(M::[[any()]]) -> M1::[[any()]].
    %% M1 az M n*n-es négyzetes mátrix olyan (n/2)*(n/2) méretű részmátrixa,
    %% mely az n/4+1. sor n/4+1. oszlopának elemétől kezdődik.
    M=[[a,b,e,f],
       [c,d,g,h],
       [i,j,m,n],
       [k,l,o,p]],     kozepe(M) =:= [[d,g],[j,m]].
       a) Használja fel listanézetben a lists:sublist/3 függvényt:
          -spec sublist(L1::list(), S::int(), L::int()) -> L2::list().
       b) Használja fel listanézetben az lists:nth/2 függvényt:
          -spec nth(N::int(), List::list()) -> Elem::any().
+9.  Mátrix középső elemei, valósítsa meg többféleképpen is
     -spec laposkozepe((M::[[any()]]) -> L::[any()].
     %% L lista az M mátrix közepe elemeinek listája.
     laposkozepe(M) =:= [d,g,j,m].
       a) flatten/1 és kozepe/1 felhasználásával.
       b) Használja fel listanézetben az lists:nth/2 függvényt.
+10. Részmátrix sor és oszlop elhagyásával
     -spec pivot((M::[[any()]], R::int(), C::int()) -> M1::[[any()]].
     %% M1 az M mátrix R. sorának és C. oszlopának elhagyásával keletkezik.
     pivot(M,2,3) =:= [[a,b,f],[i,j,n],[k,l,p]].
     A megoldáshoz használja fel listanézetben az lists:nth/2 függvényt.
+11. Lista összes nemüres részlistáját tartalmazó lista
     -spec sublists(Xs::[any()]) -> [{B::integer(), Ps::[any()], A::integer()}].
     %% Az Xs lista egy olyan (folytonos), nemüres részlistája a
     %% Ps lista, amely előtt B és amely után A számú elem áll Xs-ben.
     sublists([a,b]) =:= [{0,[a],1}, {1,[b],0}, {0,[a,b],0}].

------ $LastChangedDate: 2020-09-26 15:40:40 +0200 (szo, 26 szept 2020) $ --------
```