```
                    Deklaratív programozás, 6. gyakorlat
                    Erlang programozás: listák használata
                            2010. 11. 22. & 24.

--------------------------------------------------------------------------------

Írjon olyan Erlang-eljárást, amely megfelel az adott fejkommentnek.  Az
eljárásban felhasználhatja a már megoldott feladatokhoz megírt eljárásait.


--------------------------------------------------------------------------------
1. Listában párosával elõforduló elemek listája

%% @spec parban(Xs::[term()]) -> Zs::[term()].
%% A Zs lista az Xs lista összes olyan elemét tartalmazza, amelyet
%% vele azonos értékû elem követ.

%% Alternatív megoldás: parban/1 listanézettel.

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


--------------------------------------------------------------------------------
2. Lista egyre rövidülõ szuffixumainak listája

%% @spec tails(Ls::[term()]) -> Zss::[[term()]].
%% A Zss lista az Ls listát és egyre rövidülő szuffixumait tartalmazza.

%% dp10s-gy3.txt-ben van hasonló feladat listaSzuffixumai néven.

%% Alternatív megoldás: tails/1 foldr-rel.

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],[]].


--------------------------------------------------------------------------------
3. Listában párosával elõforduló részlisták listája

%% @spec dadogo(Ls::[term()]) -> Dss::[[term()]].
%% A Dss lista az Ls lista összes olyan nemüres (folytonos) részlistáját
%% tartalmazza, amelyet vele azonos értékû részlista követ.

dadogo([a,a,a,2,3,3,a,2,b,b,b,b]) == [[a],[a],[3],[b],[b,b],[b],[b]].
dadogo([1,2,1,2,c,a,a,c]) == [[1,2],[a]].


--------------------------------------------------------------------------------
4. Lista elsõ monoton növekvõ részlistája (futama)

%% @spec rampa(Xs::[term()]) -> {Zs::[term()], Ms::[term()]}.
%% A Zs lista az Xs lista elsõ monoton növekvõ futama, az Ms az Xs maradéka.

%%%% Segédfüggvény gyûjtõargumentummal.
%% @spec rampa(Xs::[term()], Rs::[term()]) -> {Zs::[term()],Ms::[term()]}.
%% A Zs lista az Xs lista elsõ monoton növekvõ futama az Rs fordítottja
%% mögé fûzve; az Ms az Xs maradéka (Zs utáni része).

rampa([1,2,2,3,2,4,5,6,6,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1]) ==
        {[1,2,2,3],[2,4,5,6,6,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1]}.
rampa([a,b,b,c,b,b,a]) == {[a,b,b,c],[b,b,a]}.
```



```
--------------------------------------------------------------------------------
5. Lista monoton növekvõ részlistáinak (futamainak) listája

% @spec rampak(Xs::[term()]) -> Xss::[[term()]].
% Az Xss az Xs monoton növekvõ futamainak listája.

%% Alternatív megoldás: rampak/1 slash függvénnyel.

rampak([1,2,2,3,2,4,5,6,6,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1]) ==
  [[1,2,2,3],[2,4,5,6,6,6,7],[6,8],[2,3,3,4,5,6],[0,6],[5],[4],[3],[2],[1]].


--------------------------------------------------------------------------------
6. Lista darabokra szabdalása

%% @spec slash(F::fun([term()]) -> {term(),[term()]}, Xs::([term()]))
%%                                                    -> Zs::([term()]).
%% A Zs lista az Xs listából az F ismételt alkalmazásával elõálló lista,
%% ahol F párban visszaadja a Zs következõ elemét és az Xs még nem
%% feldolgozott részét.

slash(fun(Xs) -> lists:split(3, Xs) end, "jaaaj!!! nem jooo!") ==
  ["jaa","aj!","!! ","nem"," jo","oo!"].


--------------------------------------------------------------------------------
7. 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 gyûjtõargumentummal.
%% @spec dif(Ls::[number()],N::number(),Rs::[number()]) ->
%%                                   {Ds::[number()], Zs::[number()]}.
%% A Ds lista az Ls lista N különbségû számtani sorozatot alkotó prefixuma
%% az Rs lista fordítottja mögé fûzve, a Zs az Ls maradéka (Zs utáni része).

%% Alternatív megoldás: dif/1 feltétel helyett mintaillesztéssel.

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]}.


--------------------------------------------------------------------------------
8. Lista számtani sorozatot alkotó részlistáinak listája

%% @spec difek(Xs::[number()]) -> Dss::[[number()]].
%% A Dss lista az Xs lista számtani sorozatot alkotó részlistáinak listája.

%% Alternatív megoldás: difek/1 slash függvénnyel.

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]].

%% Feladat: úgy módosítani dif/3-at, hogy ha egy szám az egyik
%% számtani sorozat utolsó és egyben a következõ elsõ eleme is
%% lehet, akkor mindkettõben vegyük figyelembe
%% NB. Ilyenkor egy sorozat záróeleme mindenképpen egy következõ
%% sorozat kezdõeleme 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]].


-----------------------------------------------------------------------------
9. Keresõfa adott tulajdonságú csomópontjainak listája

%% @type btree() = e | {n,int(),btree(),btree()}.

%%%% 9/a
%% @spec nagyobbak(T::btree(), E::int()) -> Rs::[btree()].
%% A T keresõfa-tulajdonságú fa E-nél nagyobb értéket tartalmazó
%% csomópontjainak listája Rs.

%%%% Segédfüggvény.
%% @spec s(T::btree()) -> Rs::[btree()].
%% A T keresõfa-tulajdonságú fa csomópontjainak listája Rs.

%%                   3
%%              2           6
%%            1   e       4   9
%%          e e         e e e e

nagyobbak({n,3,{n,2,{n,1,e,e},e},{n,6,{n,4,e,e},{n,9,e,e}}},2) ==
  [{n,3,{n,2,{n,1,e,e},e},{n,6,{n,4,e,e},{n,9,e,e}}},
   {n,4,e,e},
   {n,6,{n,4,e,e},{n,6,e,e}},
   {n,9,e,e}].

%%%% 9/b
%% @spec nagyobbak_utja(T::btree(), E::int()) -> Rs::[btree()]
%% Az Rs {V, Bs} párokból álló lista, ahol V a T bináris fa valamely
%% csomópontjában található, E-nél nagyobb érték, Bs pedig a V érték
%% elérési útvonalát a b(alra), illetve j(obbra) atomokkal jelzõ lista.

%%                   0
%%              1           9
%%            3   e       4   6
%%          e e         e e e e

nagyobbak_utja({n,0,{n,1,{n,3,e,e},e},
                    {n,9,{n,4,e,e},{n,6,e,e}}},2) ==
  [{3,[b,b]},{9,[j]},{4,[j,b]},{6,[j,j]}].

%%                   0
%%              1             9
%%            3   e       4     6
%%          e e         e e   7   8
%%                            e e e e

nagyobbak_utja({n,0,{n,1,{n,3,e,e},e},
                    {n,9,{n,4,e,e},{n,6,{n,7,e,e},{n,8,e,e}}}},2) ==
  [{3,[b,b]},{9,[j]},{4,[j,b]},{6,[j,j]},{7,[j,j,b]},{8,[j,j,j]}].
```



```
============================================================================
Egyszerûbb szorgalmi feladatok otthonra
============================================================================

H1. Beszúrás listába adott helyre

    %% @spec insert_nth(Ls::[term()], E::term(), N::integer()) -> Rs: [term()].

    %% Az Rs lista az Ls lista olyan másolata, amelybe az Ls lista N-edik és
    %% (N+1)-edik eleme közé be van szúrva az E elem (a lista számozása 1-tõl
    %% kezdõdik).

    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].


H2. Beszúrás rendezett listába

    %% @spec insert_ord(S0s::[integer()], E::integer()) -> Ss::[integer()].

    %% Az Ss szigorúan monoton növekvõ egészlista az S0s szigorúan monoton
    %% növekvõ egészlistának az E egésszel bõvített változata, feltéve hogy
    %% 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].


H3. Adott lista adott sorszámú eleme

    %% @spec nth1_1(N::integer(), Ls::[term()]) -> E::term().
    %% Az Ls lista N-edik eleme E (1-tol számozva az elemeket).

    Az eljárás nevében szereplo elso 1-es arra utal, hogy 1-tol számozzuk az
    elemeket, a második 1-es pedig arra, hogy ez az elso változata egy
    feladatsornak.

    nth1_1(3, [a,b,c]) == c.


H4. Adott lista sorszámozott elemeibõl álló lista

    %% @spec nth1_2(Ls::term()) -> {N::integer(), E::term()}.
    %% Az Ls lista N-edik eleme E (1-tol számozva az elemeket).

    nth1_2([a,b,c]) == [{1,a},{2,b},{3,c}].


H5. Lista adott hosszúságú prefixuma

    Egy N elemu Xs lista prefixumának nevezzük az L hosszságú Ps listát, ha a
    Ps az Xs elso L elemébõl áll (az Xs-beli sorrend megtartásával), ahol 0 =<
    L =< N.

    %% @spec prefix_length(L::integer(), Xs::[term()]) -> Ps::[term()].
    %% Az Xs lista L hosszúságú prefixuma a Ps lista.

    prefix_length(3, [a,b,c,d,e]) == [a,b,c].

    Megjegyzés: a prefix_length-szel azonos funkcionalitású függvények szokásos
    neve a funkcionális nyelvekben: take.
```

```
H6. Lista adott helyen kezdodo szuffixuma

    Egy N elemu Xs lista szuffixumának nevezzük az N-B hosszúságú Ss listát, ha
    az Ss az Xs első B eleme utáni elemekből áll (az Xs-beli sorrend
    megtartásával), ahol 0 =< B =< N.

    %% @spec suffix_before(B::integer(), Xs::[term()]) -> Ss::[term()].
    %% Az Xs lista olyan szuffixuma Ss, amely az Xs első B elemét nem
    %% tartalmazza.

    suffix_before(3, [a,b,c,d,e]) == [d,e].

    Megjegyzés: a suffix_before-ral azonos funkcionalitású függvények szokásos
    neve a funkcionális nyelvekben: drop.


H7. Részlista képzése

    %% sublist(B::integer(), L::integer(), Xs::[term()]) -> Ps::[term()].
    %% Az Xs lista olyan (folytonos) részlistája az L hosszúságú Ps lista,
    %% amely elott B számú elem áll Xs-ben.

    sublist(1, 3, [a,b,c,d,e]) == [b,c,d].


H8. Lista adott hosszúságú összes részlistáját tartalmazó lista

    %% sublist(L::integer(), Xs::[term()]) ->
    %%                           [{B::integer(), Ps::[term()], A::integer()}].
    %% Az Xs lista egy olyan (folytonos) részlistája az L hosszúságú Ps lista,
    %% amely elott B és amely után A számú elem áll Xs-ben.

    sublist(1,[a,b,c]) == [{0,[a],2}, {1,[b],1}, {2,[c],0}].
    sublist(2,[a,b,c]) == [{0,[a,b],1}, {1,[b,c],0}].


H9. Lista összes nemüres részlistáját tartalmazó lista

    %% sublist(Xs::[term()]) -> [{B::integer(), Ps::[term()], A::integer()}].
    %% Az Xs lista egy olyan (folytonos), nemüres részlistája a Ps lista, amely
    %% elott B és amely után A számú elem áll Xs-ben.

    sublist(1,[a,b]) == [{0,[a],1}, {1,[b],0}, {0,[a,b],0}].
```