```
-module('dp20a-gy2').
-compile(export_all).
-author('patai@iit.bme.hu, hanak@iit.bme.hu, kapolnai@iit.bme.hu').
-vsn('$LastChangedDate: 2020-10-09 12:59:23 +0200 (p, 09 okt 2020) $$').

%-------------------------------------------------------------------------
%                            LISTA
%-------------------------------------------------------------------------

% 1.
-spec seq(F::integer(), T::integer()) -> S::[integer()].
%% S = [F, F+1, ..., T].
seq(F, T) ->
    seq(F, T, []).

-spec seq(F::integer(), T::integer(), L::[integer()]) -> S::[integer()].
%% seq(F, T, L) = F..T sorozat L elé fűzve.
seq(F, T, L) when T<F -> L;
seq(F, T, L) ->
    seq(F, T-1, [T|L]).

%-------------------------------------------------------------------------

% 2.
-spec all_different(Xs::[any()]) -> B::boolean().
%% B igaz, ha az L listában csupa különböző értékű elem van.
all_different([]) ->
    true;
all_different([H|T]) ->
    not lists:member(H, T) andalso all_different(T).

-spec all_different2(Xs::[any()]) -> B::boolean().
all_different2(L) -> length(L) =:= length(lists:usort(L)).

%-------------------------------------------------------------------------

% 3.
-spec van2eleme(L::list()) -> R::boolean().
%% R =:= length(L) >= 2.
van2eleme([_,_|_]) -> true;
van2eleme(_) -> false.

%-------------------------------------------------------------------------

% 4.
-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(L, S, N) ->
    [lists:nth(I, L) || I <- lists:seq(S, S+N-1)].

%-------------------------------------------------------------------------

% 5.
-spec all(P::fun((T::any()) -> boolean()), L::[T::any()]) -> B::boolean().
all(Pred, L) ->
    lists:foldl(fun(X,E) -> X andalso E end, true, lists:map(Pred, L)).

-spec all2(P::fun((T::any()) -> boolean()), L::[T::any()]) -> B::boolean().
all2(Pred, L) ->
    lists:foldl(fun(X,E) -> Pred(X) andalso E end, true, L).

-spec all3(P::fun((T::any()) -> boolean()), L::[T::any()]) -> B::boolean().
%% B akkor és csak akkor igaz, ha Pred teljesül az L minden elemére.
all3(Pred, [Hd|Tail]) ->
    case Pred(Hd) of
        true -> all3(Pred, Tail);
        false -> false
    end;
all3(Pred, []) when is_function(Pred, 1) ->
    true.
```

```
-spec all4(P::fun((T::any()) -> boolean()), L::[T::any()]) -> B::boolean().
all4(_Pred, []) ->
    true;
all4(Pred, [H|T]) ->
    Pred(H) andalso all4(Pred, T).

%-------------------------------------------------------------------------

% 6.
-spec platok_hossza(Xs::[any()]) -> PHs::[{P::any(), H::integer()}].
%% Platónak nevezzük az egyenlő értékű elemek sorozatát egy listában.
%% PHs olyan {P, H} párok listája, amelyekben P a platót képező
%% érték, H pedig e plató hossza (= azonos értékű elemeinek száma).
platok_hossza([])->
    [];
platok_hossza(Xs) ->
    {[P|_Ps] = PPs, Ss} = plato(Xs),
    [ {P, length(PPs)} | platok_hossza(Ss) ].

-spec plato(Xs::[any()]) -> PsMs::{Ps::[any()], Ms::[any()]}.
%% PsMs egy olyan {Ps, Ms} pár, amelyben Ps az Xs lista azonos értékű
%% elemekből álló, tovább már nem bővíthető prefixuma, Ms pedig az Xs
%% további elemeinek listája.
plato([X|[X|_Ys] = XYs]) ->
    {Ps, Ms} = plato(XYs),
    {[X|Ps], Ms};
plato([X|[_Y|_Ys] = YYs]) ->
    {[X], YYs};
plato(Xs) -> % itt már csak egy elemű vagy üres lehet az Xs lista
    {Xs, []}.

%-------------------------------------------------------------------------

% 7.
-spec duplak(Xs::[any()]) -> Ys::[any()].
%% Ys az Xs azon elemeinek listája, melyek azonosak az őket követő elemmel.
duplak([]) -> [];
duplak(L) ->
    [ E || {E,E} <- lists:zip(lists:sublist(L, length(L)-1), tl(L)) ].

%-------------------------------------------------------------------------

% 8.
-spec kozepe(M::list([any()])) -> M1::list([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.
kozepe(M) ->
    N = length(M),
    N4 = N div 4,
    N2 = N div 2,
    [ lists:sublist(R, N4 + 1, N2) || R <- lists:sublist(M, N4 + 1, N2) ].

-spec kozepe2(M::list([any()])) -> M1::list([any()]).
kozepe2(M) ->
    N = length(M),
    Seq = lists:seq(N div 4 + 1, N div 4 + N div 2),
    [ [ lists:nth(C, lists:nth(R, M)) || C <- Seq ]
      || R <- Seq
    ].

%-------------------------------------------------------------------------

% 9.
-spec laposkozepe(M::list([any()])) -> L::[any()].
%% Az L lista az M mátrix középső elemeinek listája.
laposkozepe(M) ->
    lists:flatten(kozepe(M)).
```

```
-spec laposkozepe2(M::list([any()])) -> L::[any()].
laposkozepe2(M) ->
    N = length(M),
    Seq = lists:seq(N div 4 + 1, N div 4 + N div 2),
    [ lists:nth(C, lists:nth(R, M))
      || R <- Seq,
         C <- Seq
    ].

%-------------------------------------------------------------------------

% 10.
-spec pivot(M::list([any()]),R::integer(),C::integer()) -> M1::list([any()]).
%% M1 az M mátrix R-edik sorának és C-edik oszlopának elhagyásával áll elő.
pivot(M, R, C) ->
    N = length(M),
    [ [ lists:nth(Co, lists:nth(Ro, M)) || Co <- lists:seq(1, N), Co =/= C ]
      || Ro <- lists:seq(1, N), Ro =/= R
    ].

%-------------------------------------------------------------------------

% 11.
-spec transpose(M::list([any()])) -> MT::list([any()]).
%% Az M mátrix transzponáltja, azaz a sorainak és oszlopainak
%% felcserélésével előálló mátrix MT.
%% Megjegyzés: egy mátrix minden sora egyforma hosszú.
%% Az alábbi transpose és transpose2 függvényekben feltételezzük, hogy a
%% bemeneti mátrix helyes.
%% A transpose3 variáns -- a listanézet használata miatt -- akkor is működik,
%% ha a bemeneti mátrix sorai különböző hosszúságúak: ha egy sor rövidebb a
%% leghosszabbnál, akkor abból a sorból az utolsó oszlop(ok)ba nem kerül(nek)
%% be elem(ek).

transpose([]) ->
    [];
transpose([[] | _Matrix]) ->
    [];
transpose(Matrix) ->
    [lists:map(fun hd/1, Matrix) | transpose(lists:map(fun tl/1, Matrix))].

-spec transpose2(M::list([any()])) -> MT::list([any()]).
%% Szeredi Péter megoldása
transpose2([]) ->
    [];
transpose2([[] | _Matrix]) ->
    [];
transpose2(Matrix) ->
    {FirstCol,Rest} = splice_matrix(Matrix),
    [FirstCol|transpose2(Rest)].

-spec splice_matrix(Matrix::list([any()])) -> FirstRest::{FirstCol::list(any()),RestMx::l
ist([any()])}.
splice_matrix([]) ->
    {[],[]};
splice_matrix([[H1|Row1]|Matrix]) ->
    {Col1,Rest} = splice_matrix(Matrix),
    {[H1|Col1], [Row1|Rest]}.

-spec transpose3(M::list([any()])) -> MT::list([any()]).
transpose3([]) ->
    [];
transpose3([[] | _Matrix]) ->
    [];
transpose3(Matrix) ->
    [[H || [H|_] <- Matrix] | transpose3([T || [_|T] <- Matrix])].
```

```
%-----------------------------------------------------------------------------

% +1.
-spec osszetett(K::integer()) -> L::[integer()].
%% L tartalmazza az összetett számokat 4..K*K között, ismétlődés lehet.
osszetett(K) ->
   % lists:usort
([ J || I <- lists:seq(2,K), J <- lists:seq(I*2, K*K, I)]).

%-----------------------------------------------------------------------------

% +2.
-spec primek(K::integer()) -> L::[integer()].
%% L tartalmazza a prímszámokat 2..K*K között.
primek(K) ->
    Os = osszetett(K),
    [ X || X <- lists:seq(2,K*K), not lists:member(X,Os)].

%-----------------------------------------------------------------------------

% +3.
-spec zip(Xs::[any()], Ys::[any()]) -> XYs::[{any(), any()}].
%% XYs olyan párok listája, amelyek első eleme az Xs, második eleme
%% az Ys lista azonos pozíciójú eleme.
zip([], []) ->
    [];
zip([H1|T1], [H2|T2]) ->
    [{H1,H2}|zip(T1,T2)].

-spec unzip(XYs::[{any(), any()}]) -> {Xs::[any()], Ys::[any()]}.
%% XYs olyan párok listája, amelyek első eleme az Xs, második eleme
%% az Ys lista azonos pozíciójú eleme.
unzip([]) ->
    {[],[]};
unzip([{H1,H2}|T]) ->
    {T1,T2} = unzip(T),
    {[H1|T1],[H2|T2]}.
```

```
%-------------------------------------------------------------------------

test(lista) ->
    M=[[a,b,e,f],
       [c,d,g,h],
       [i,j,m,n],
       [k,l,o,p]],
     M0 = [[a,b],
           [c,d],
           [e,f]],
     M1 = [[a,e,i,m],
           [b,f,j,n],
           [c,g,k,o],
           [d,h,l,p]],
     M2 = [[a,e,i,m],
           [b,f,j],
           [c,g],
           [d,h,l,p]],
    io:format("~p~n",
    [[
     seq(10,13) =:= [10,11,12,13],
%     flatten([1,[2,3],[[[[4]]],[5,[6]]]]) =:= [1,2,3,4,5,6],
     all_different([1,2,3,1]) =:= false,
     all_different([1,2,3]) =:= true,
     all_different2([1,2,3,1]) =:= false,
     all_different2([1,2,3]) =:= true,
     van2eleme([]) =:= false,
     van2eleme([a]) =:= false,
     van2eleme([a,b]) =:= true,
     van2eleme([a,b,c]) =:= true,
     duplak([1,2,3]) =:= [],
     duplak([1,1,2,3,3,3]) =:= [1,3,3],
     all(fun is_atom/1, [a,b,c]) and not all(fun is_atom/1, [a,b,1]),
     all2(fun is_atom/1, [a,b,c]) and not all(fun is_atom/1, [a,b,1]),
     all3(fun is_atom/1, [a,b,c]) and not all(fun is_atom/1, [a,b,1]),
     all4(fun is_atom/1, [a,b,c]) and not all(fun is_atom/1, [a,b,1]),
     platok_hossza([a,a,a,b,b,c,c,c,c,d,e,f,f,f,g,h]) =:=
         [{a,3},{b,2},{c,4},{d,1},{e,1},{f,3},{g,1},{h,1}],
     platok_hossza([a,a,a,b,b,d,f,f,h]) =:= [{a,3},{b,2},{d,1},{f,2},{h,1}],
     sublist([1,2,3,4],2,2) =:= [2,3],
     sublist([1,2,3,4],2,3) =:= [2,3,4],
     kozepe(M) =:= [[d,g],[j,m]],
     kozepe2(M) =:= [[d,g],[j,m]],
     laposkozepe(M) =:= [d,g,j,m],
     laposkozepe2(M) =:= [d,g,j,m],
     pivot(M,2,3) =:= [[a,b,f],[i,j,n],[k,l,p]],
     osszetett(5) =:= [4,6,8,10,12,14,16,18,20,22,24,6,9,
                       12,15,18,21,24,8,12,16,20,24,10,15,20,25],
     primek(5) =:= [2,3,5,7,11,13,17,19,23],
     zip([1,2,3], [a,b,c]) =:= [{1,a},{2,b},{3,c}],
     unzip([{1,a},{2,b},{3,c}]) =:= {[1,2,3], [a,b,c]},
     transpose(M0) =:= [[a,c,e], [b,d,f]],
     transpose(M1) =:= [[a,b,c,d], [e,f,g,h], [i,j,k,l], [m,n,o,p]],
     try transpose(M2) catch error:_ -> "nem mátrix" end =:= "nem mátrix",
     transpose2(M0) =:= [[a,c,e], [b,d,f]],
     transpose2(M1) =:= [[a,b,c,d], [e,f,g,h], [i,j,k,l], [m,n,o,p]],
     try transpose2(M2) catch error:_ -> "nem mátrix" end =:= "nem mátrix",
     transpose3(M0) =:= [[a,c,e], [b,d,f]],
     transpose3(M1) =:= [[a,b,c,d], [e,f,g,h], [i,j,k,l], [m,n,o,p]],
     transpose3(M2) =:= [[a,b,c,d], [e,f,g,h], [i,j,l], [m,p]]
    ]]).
```