```erlang
-module(dp12a_gy3).
-compile(export_all).
-author('patai@iit.bme.hu, hanak@iit.bme.hu, kapolnai@iit.bme.hu').
-vsn('$LastChangedDate: 2012-10-09 18:52:19 +0200 (Tue, 09 Oct 2012) $$').

% 1.

lista_noveltje_1([])    -> [];
lista_noveltje_1([H|T]) -> [H+1|lista_noveltje_1(T)].

% 2.

last_1([X])   -> X;
last_1([_|T]) -> last_1(T).

% 3.

safe_last([_|_] = Xs) -> {ok, last_1(Xs)};
safe_last(_)          -> error.

% 4.

split(0, L) ->
    {[], L};
split(N, [H|T]) ->
    {PT,S} = split(N-1, T),
    {[H|PT], S}.

% 5.

take(L0, N) ->
    element(1, split(N, L0)).

% 6.

drop(L0, N) ->
    element(2, split(N, L0)).

% 7.

prefixes_1(Xs) ->
    prefixes(Xs, 0).

%% prefixes(Xs::[term()], N::integer()) -> Xss::[[term()]].
%% Xss az Xs listának összes, legalább N hosszú prefixuma.
prefixes(Xs, N) ->
    case length(Xs) >= N of
        true  -> [take(Xs, N)|prefixes(Xs, N+1)];
        false -> []
    end.

% vagy őrrel:
prefixes_b(Xs, N) when length(Xs) >= N ->
    [take(Xs, N)|prefixes_b(Xs, N+1)];
prefixes_b(_Xs, _N) ->
    [].

% 8.

tails_1([_H|T]=Xs) ->  [Xs|tails_1(T)];
tails_1([]) ->         [[]].

% 9.

sublists(N, Xs) ->
    sublists(N, 0, Xs).

sublists(N, B, Xs) ->
    case length(Xs) < N of
        true  -> [];
        false -> [{B, take(Xs, N), length(Xs) - N} | sublists(N, B+1, tl(Xs))
]
    end.

% 10.

sublists_1(Xs) ->
    sublists_atleast(1, Xs).

%% sublists_atleast(N::integer(), Xs::[term()]) -> Xss::[[term()]].
%% Xss az Xs legalább N hosszú részlistáinak listája.
sublists_atleast(N, Xs) ->
    case length(Xs) < N of
        true  -> [];
        false -> sublists(N, Xs) ++ sublists_atleast(N+1, Xs)
    end.

% 11.

parban_1([E,E|T])    -> [E|parban_1([E|T])];
parban_1([_E1,E2|T]) -> parban_1([E2|T]);
parban_1(_)          -> [].

% 12.

cons(H, T) ->
    [H|T].

append(Xs, Ys) ->
    lists:foldr(fun cons/2, Ys, Xs).

% 13.

revapp(Xs, Ys) ->
    lists:foldl(fun cons/2, Ys, Xs).

% 14.

tails_2(Xs) ->
    lists:foldr(fun(Y, [Zs|Zss]) -> [[Y|Zs],Zs|Zss] end, [[]], Xs).

% 15.

map(F, L) ->
    [F(X) || X <- L].

filter(F, L) ->
    [X || X <- L, F(X)].

% 16.

lista_noveltje_2(L0) ->
    [X+1 || X <- L0].

% 17.

tails_3(Xs) ->
    [drop(Xs, N) || N <- lists:seq(0, length(Xs))].

% 18.

sublists_2(Xs) ->
    [{B,take(drop(Xs, B), N),length(Xs)-B-N} || N <- lists:seq(1, length(Xs))
,
                                                B <- lists:seq(0, length(Xs)-
N)].

% vagy:

sublists_2b(Xs) ->
```

```
    lists:flatten([sublists(N, Xs) || N <- lists:seq(1, length(Xs))]).

% 19.

parban_2(Xs) ->
    [E || [E,E|_] <- tails_1(Xs)].

% 20.

dadogo(Xs) ->
    [P || T <- tails_1(Xs),
          N <- lists:seq(1, length(T) div 2),
          begin {P,S} = split(N, T), P =:= take(S, N) end
    ].

% 21.

rampa_1(L) ->
    case L of
        [X1,X2|Xs] -> case X1 =< X2 of
                          true  -> {Zs,Ms} = rampa_1([X2|Xs]),
                                   {[X1|Zs], Ms};
                          false -> {[X1], [X2|Xs]}
                      end;
        L          -> {L, []}  % vagyis ha length(L) < 2
    end.

% vagy őrrel:

rampa_1b([]) ->  % ha ures listaval hivjak meg
    {[], []};
rampa_1b([X1,X2|Xs]) when X1 =< X2 ->
    {Zs,Ms} = rampa_1b([X2|Xs]),
    {[X1|Zs], Ms};
rampa_1b([X|Xs]) ->
    {[X], Xs}.

% lásd http://www.erlang.org/doc/man/lists.html

rampa_2([]) ->
    {[],[]};
rampa_2(Ls) ->
    Rs = lists:takewhile(fun({X,Y}) -> X =< Y end,
                         lists:zip(lists:sublist(Ls, length(Ls)-1), tl(Ls))),
    lists:split(length(Rs)+1, Ls).

%% Alternatív megoldás tails/1 és lists:splitwith/2 használatával.
%%
%% rampa_3/1-ben tails/1 eredménye az Ls egyre rövidülo farkainak a listája, pl.
%% tails([a,b,c,b,a]) =:= [a,b,c,b,a]
%%                          [b,c,b,a]
%%                            [c,b,a]
%%                              [b,a]
%%                                [a]
%%                                  []
%% splitwith/2 közülük azokat adja vissza Rss-ben, amelyeknek a feje nem nagyobb
%% a második elemüknél, Mss-ben pedig - az elso kivételével - azokat amelyekre
%% ez nem áll fenn, pl.
%%    Rss =:= [[a,b,c,b,a],[b,c,b,a]]
%%    Mss =:= [[b,a],[a],[]]
%%
%% Ha ezek után az Rss lista üres, akkor egyedül Ls feje "képez" monoton növekvo
%% sorozatot, az Ls farka pedig a maradék.
%%
%% Ha az Rss nem üres, akkor elso részlistájának a fejébol és összes
%% részlistájának a második elemébol képezzük a monoton növekvo sorozatot (az
%% a futamot, az eredménypár elso tagját), az Mss minden nemüres részlistáján az
%% az elso elemébol pedig a maradéklistát (az eredménypár második tagját).
rampa_3([]) ->
    {[],[]};
rampa_3(Ls) ->
    F = fun([X1,X2|_]) -> X1 =< X2; (_) -> false end,
    {Rss,[_|Mss]} = lists:splitwith(F, tails_1(Ls)),
    case Rss of
        [] ->
            lists:split(1, Ls);
        [Es|_] ->
            % {futam (eredménypár 1. tagja), maradéklista (eredménypár 2. tagja}
            {[hd(Es)|[X || [_,X|_] <- Rss]], [X || [X|_] <- Mss]}
    end.


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


t() ->
    Paros = fun(X) -> X rem 2 =:= 0 end,
    Xs = [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],
    Zs = [1,2,2,3],
    Ms = [2,4,5,6,6,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1],
    (lista_noveltje_1([1,5,2]) =:= [2,6,3])
        and (last_1([5,1,2,8,7]) =:= 7)
        and (safe_last([5,1,2,8,7]) =:= {ok,7})
        and (safe_last([]) =:= error)
        and (split(3, [a,b,c,d,e]) =:= {[a,b,c],[d,e]})
        and (take([10,20,30,40,50], 3) =:= [10,20,30])
        and (drop([10,20,30,40,50], 3) =:= [40,50])
        and (prefixes_1([a,b,c]) =:= [[], [a], [a,b], [a,b,c]])
        and (tails_1([1,4,2]) =:= [[1,4,2],[4,2],[2],[]])
        and (sublists(1,[a,b,c]) =:= [{0,[a],2}, {1,[b],1}, {2,[c],0}])
        and (sublists(2,[a,b,c]) =:= [{0,[a,b],1}, {1,[b,c],0}])
        and (sublists_1([a,b]) =:= [{0,[a],1}, {1,[b],0}, {0,[a,b],0}])
        and (parban_1([a,a,a,2,3,3,a,2,b,b,4,4]) =:= [a,a,3,b,4])
        and (append([a,b,c], [1,2,3]) =:= [a,b,c,1,2,3])
        and (revapp([a,b,c], [1,2,3]) =:= [c,b,a,1,2,3])
        and (tails_2([1,4,2]) =:= [[1,4,2],[4,2],[2],[]])
        and (map(Paros, [1,2,3,4]) =:= [false,true,false,true])
        and (filter(Paros, [1,2,3,4]) =:= [2,4])
        and (lista_noveltje_2([1,5,2]) =:= [2,6,3])
        and (tails_3([1,4,2]) =:= [[1,4,2],[4,2],[2],[]])
        and (sublists_2([a,b]) =:= [{0,[a],1}, {1,[b],0}, {0,[a,b],0}])
        and (sublists_2b([a,b]) =:= [{0,[a],1}, {1,[b],0}, {0,[a,b],0}])
        and (parban_2([a,a,a,2,3,3,a,2,b,b,4,4]) =:= [a,a,3,b,4])
        and (dadogo([a,a,a,2,3,3,a,b,b,b,b]) =:= [[a],[a],[3],[b],[b,b],[b],[b]])
        and (rampa_1(Xs) =:= {Zs, Ms})
        and (rampa_2(Xs) =:= {Zs, Ms})
        and (rampa_3(Xs) =:= {Zs, Ms})
        and true.
```