```
-module(dp12a_gy4).
-compile(export_all).
-author('patai@iit.bme.hu, hanak@iit.bme.hu, kapolnai@iit.bme.hu').
-vsn('$LastChangedDate: 2012-10-29 21:39:15 +0100 (Mon, 29 Oct 2012) $$').

%%------------------------------------------------------------------------------
%%                           I. RÉSZ: FÁK
%%------------------------------------------------------------------------------

% 1.
fa_noveltje(level) ->
    level;
fa_noveltje({C,Bfa,Jfa}) ->
    {C+1, fa_noveltje(Bfa), fa_noveltje(Jfa)}.

% 2.
fa_tukorkepe(level) ->
    level;
fa_tukorkepe({C,Bfa,Jfa}) ->
    {C, fa_tukorkepe(Jfa), fa_tukorkepe(Bfa)}.

% 3. a)
fa_balerteke(level) ->
    error;
fa_balerteke({C,level,_}) ->
    {ok, C};
fa_balerteke({_,Bfa,_}) ->
    fa_balerteke(Bfa).

% 3. b)
fa_jobberteke(level) ->
    error;
fa_jobberteke({C,_,level}) ->
    {ok, C};
fa_jobberteke({_,_,Jfa}) ->
    fa_jobberteke(Jfa).

% 4.
rendezett_fa(level) ->
    true;
rendezett_fa({C,Bfa,Jfa}) ->
    case fa_jobberteke(Bfa) of
        error  -> true;
        {ok,J} -> J < C
    end andalso
        rendezett_fa(Bfa) andalso
    case fa_balerteke(Jfa) of
        error  -> true;
        {ok,B} -> C < B
    end andalso
        rendezett_fa(Jfa).

% 4. kicsit hatékonyabb változatokban a fájl végén, kiegészítő anyag

% 5.
tartalmaz(_, level) ->
    false;
tartalmaz(C, {C,_,_}) ->
    true;
tartalmaz(C, {_,Bfa,Jfa}) ->
    tartalmaz(C, Bfa) orelse tartalmaz(C, Jfa).

% 6.
elofordul(_, level) ->
    0;
elofordul(C, {R,Bfa,Jfa}) ->
    if C =:= R -> 1;
       true    -> 0
    end
        + elofordul(C, Bfa) + elofordul(C, Jfa).

% 7.
utak(Fa) -> utak(Fa, []).

% @spec utak(F::fa(), Eddigi::ut()) -> CimkezettUtak::[{C::term(), U::ut()}].
% A CimkezettUtak lista az F fa minden csomópontjához egy kételemű ennest
% társít, amelynek első eleme (C) a csp. címkéje, második eleme (U) az
% Eddigi útvonal és a csp. útvonala összefűzve.
utak(level, _) ->
    [];
utak({C,Bfa,Jfa}, Eddigi) ->
    Eddigi1 = Eddigi ++ [C], % Költséges művelet!
    [{C, Eddigi} | utak(Bfa, Eddigi1)] ++ utak(Jfa, Eddigi1).

% 8. a)
cutak_a(C, Fa) ->
    [CU || {C0,_} = CU <- utak(Fa), C0 =:= C].

% 8. b)
cutak_b(C, Fa) -> cutak(C, Fa, []).

% @spec ut(C::term(), F::fa(), Eddigi::ut()) ->
%                 CimkezettUtak::[{C::term(), U::ut()}].
% A CimkezettUtak lista az F fa minden C címkéjű csomópontjához egy kételemű
% ennest társít, amelynek első eleme C, második eleme az Eddigi útvonal és
% és a csp. útvonala összefűzve.
cutak(_, level, _) ->
    [];
cutak(C, {R,Bfa,Jfa}, Eddigi) ->
    Eddigi1 = Eddigi ++ [R],
    Cutak = cutak(C, Bfa, Eddigi1) ++ cutak(C, Jfa, Eddigi1),
    if R =:= C -> [{C, Eddigi} | Cutak];
       true    -> Cutak
    end.


%%------------------------------------------------------------------------------
%%                           II. RÉSZ: LISTÁK
%%------------------------------------------------------------------------------


% 9.
zip([], []) ->
    [];
zip([H1|T1], [H2|T2]) ->
    [{H1,H2}|zip(T1,T2)].

unzip([]) ->
    {[],[]};
unzip([{H1,H2}|T]) ->
    {T1,T2} = unzip(T),
    {[H1|T1],[H2|T2]}.

% 10.
duplak([]) -> [];
duplak(L) ->
    [ E || {E,E} <- zip(lists:sublist(L, length(L)-1), tl(L)) ].

% 11a.
all_different_a([]) ->
    true;
all_different_a([H|T]) ->
    not lists:member(H, T) andalso all_different_a(T).


% 11b.
all_different_b(L) -> length(L) =:= length(lists:usort(L)).

% 12.
descs() ->
```

```
    Ds = lists:seq(1,6),
    [ [A,B,C,D,E,F] || A <- Ds,
                       B <- Ds,
                       C <- Ds,
                       D <- Ds,
                       E <- Ds,
                       F <- Ds
    ].

% 13. Nagyon lassú megoldás:
perms_0() ->
    Ds = lists:seq(1,6),
    [ [A,B,C,D,E,F] || A <- Ds,
                       B <- Ds,
                       C <- Ds,
                       D <- Ds,
                       E <- Ds,
                       F <- Ds,
%                      L <- [[A,B,C,D,E,F]] % generátorral tudnánk kötni változóhoz értéket, hogy ne kelljen 2x felépíteni L-et
                       all_different_b([A,B,C,D,E,F])
    ].

% 13. Gyorsabb megoldás:
perms_1() ->
    Ds = lists:seq(1,6),
    [ [A,B,C,D,E,F] || A <- Ds,
                       B <- Ds -- [A],
                       C <- Ds -- [A,B],
                       D <- Ds -- [A,B,C],
                       E <- Ds -- [A,B,C,D],
                       F <- Ds -- [A,B,C,D,E]
    ].

% 13. Rugalmasabb megoldás:
perms_2() ->
    perms_2(lists:seq(1,6)).

% perms_2(L) az L lista elemeinek permutációit tartalmazó lista.
% Hátulról épít.
perms_2([X]) ->
    [[X]];
perms_2(L) ->
    [ [H|T] || H <- L, T <- perms_2(L--[H]) ].

perms_3() -> perms_3(lists:seq(1,6), []).

% perms_3(L) az L lista elemeinek permutációit tartalmazó,
% Prefix mögé fűzött lista. Elölről épít.
perms_3([], Prefix) ->
    [Prefix];
perms_3(L, Prefix) ->
    [ T || H <- L, T <- perms_3(L--[H], Prefix ++ [H]) ].


test(fa) ->
    T1 = {4,
          {3,level,level},
          {6,
           {5,level,level},
           {7,level,level}}},
    T2 = {a,
          {b, {x,level,level}, level},
          {c,
           level,
           {d,
            {x,{e,level,level},level},
            {a, {x,level,level},{x,level,level}}}}},
    T3 = {4,
          {3,
           {2,level,level},
           {1,level,level}},
          {6,
           {5,level,level},
           {7,level,level}}},
%    lists:foldl(fun erlang:'and'/2, true,
    [fa_noveltje(T1) =:= {5,{4,level,level},{7,{6,level,level},{8,level,level}}},
     fa_tukorkepe(T1) =:= {4,{6,{7,level,level},{5,level,level}},{3,level,level}},
     fa_balerteke(T1) =:= {ok, 3},
     fa_balerteke(level) =:= error,
     fa_balerteke(T2) =:= {ok, x},
     fa_jobberteke(T1) =:= {ok, 7},
     rendezett_fa(T1) =:= true,
     rendezett_fa(T2) =:= false,
     rendezett_fa(T3) =:= false,
     rendezett_fa(T1) =:= rendezett_fa_2(T1),
     rendezett_fa(T2) =:= rendezett_fa_2(T2),
     rendezett_fa(T3) =:= rendezett_fa_2(T3),
     rendezett_fa(T1) =:= rendezett_fa_3(T1),
     rendezett_fa(T2) =:= rendezett_fa_3(T2),
     rendezett_fa(T3) =:= rendezett_fa_3(T3),
     tartalmaz(x, T1) =:= false,
     tartalmaz(x, T2) =:= true,
     elofordul(x, T1) =:= 0,
     elofordul(x, T2) =:= 4,
     utak(T1) =:= [{4,[]},{3,[4]},{6,[4]},{5,[4,6]},{7,[4,6]}],
     utak(T2) =:= [{a,[]},{b,[a]},{x,[a,b]},{c,[a]},{d,[a,c]},{x,[a,c,d]},
                   {e,[a,c,d,x]},{a,[a,c,d]},{x,[a,c,d,a]},{x,[a,c,d,a]}],
     cutak_a(x,T1) =:= [],
     cutak_a(x,T2) =:= [{x,[a,b]},{x,[a,c,d]},{x,[a,c,d,a]},{x,[a,c,d,a]}],
     cutak_a(x,T1) =:= cutak_b(x,T1),
     cutak_a(x,T2) =:= cutak_b(x,T2)
    ]
%               )
        ;

test(lista) ->
    [
     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]},
     duplak([1,2,3]) =:= [],
     duplak([1,1,2,3,3,3]) =:= [1,3,3],
     all_different_a([1,3,5]) =:= true,
     all_different_a([1,3,1]) =:= false,
     all_different_b([1,3,5]) =:= true,
     all_different_b([1,3,1]) =:= false,
     begin
         _ = statistics(runtime),
         io:format("Futasi idot merek...~n"),
         DescsLength = length(descs()),
         io:format("descs ideje: ~w ms~n", [element(2, statistics(runtime))]),
         P0 = perms_0(),
         io:format("perms_0 ideje: ~w ms~n", [element(2, statistics(runtime))]),
         P1 = perms_1(),
         io:format("perms_1 ideje: ~w ms~n", [element(2, statistics(runtime))]),
         P2 = perms_2(),
         io:format("perms_2 ideje: ~w ms~n", [element(2, statistics(runtime))]),
         P3 = perms_3(),
         io:format("perms_3 ideje: ~w ms~n", [element(2, statistics(runtime))]),
         DescsLength =:= 6*6*6*6*6*6
             andalso
         length(lists:usort([ lists:usort(P) || P <- [P0,P1,P2,P3] ])) =:= 1
     end
```

```
    ].


%-----------------------------------------------------------------------------
%          KIEGÉSZÍTÉS: 4. feladat kicsit hatékonyabb változatai
%-----------------------------------------------------------------------------

% 4. hatékonyabban (bonyolult, ráadásul lehetne még javítani)
rendezett_fa_2(level) ->
    true;
rendezett_fa_2({C,Bfa,Jfa}=F) ->
    Also = case fa_balerteke(Bfa) of
               {ok, AE} -> AE;
               error -> C
           end,
    Felso = case fa_jobberteke(Jfa) of
                {ok, FE} -> FE;
                error -> C
            end,
    rendezett_fa_2_kozott(Also,Felso,F).

% rendezett_fa_2_kozott(Also::term(), Felso::term(), F::fa()) -> B::bool().
% B igaz, ha F rendezett ÉS minden C címkéjére Also =< C andalso C =< Felso,
% az = csak akkor megengedett, ha C balérték (Also) vagy jobbérték (Felso).
rendezett_fa_2_kozott(Also, Felso, {C,level,level}) ->
    Also =< C andalso C =< Felso;
rendezett_fa_2_kozott(Also, Felso, {C,Bfa,level}) ->
    Also < C andalso C =< Felso andalso
        rendezett_fa_2_kozott(Also,C,Bfa);
rendezett_fa_2_kozott(Also, Felso, {C,level,Jfa}) ->
    Also =< C andalso C < Felso andalso
        rendezett_fa_2_kozott(C,Felso,Jfa);
rendezett_fa_2_kozott(Also, Felso, {C,Bfa,Jfa}) ->
    Also < C andalso C < Felso andalso
        rendezett_fa_2_kozott(Also,C,Bfa) andalso
        rendezett_fa_2_kozott(C,Felso,Jfa).

% 4. még hatékonyabban (csak egyszer járja be).
rendezett_fa_3(Fa) ->
    case mmrendezett(Fa) of
        {_,_,R} -> R;
        R       -> R
    end.

%% @spec mmrendezett(F::fa()) ->
%%       {Min::term(), Max::term(), Rendezett::bool()} | Rendezett::bool().
%% Az F fa balértéke Min, jobbértéke Max, és rendezettsége Rendezett,
%% de olykor Min és Max nem kerül meghatározásra;
%% ezen esetek: üres fa, bizonyos nem rendezett fák.
mmrendezett(level) ->
    true;
mmrendezett({C,Bfa,Jfa}) ->
    % Első case-ben a bal fa rendezettségét vizsgáljuk,
    % ezután BR a bal fa rendezettsége, Min a balérték,
    % CBR a jobb fa elhagyásával keletkező {C,Bfa,level} fa rendezettsége.
    CBR = case mmrendezett(Bfa) of
              {BMin,BMax,BR} -> Min=BMin,
                                BR andalso BMax < C;  % CBR a case értéke
              BR             -> Min=C,
                                BR  % CBR a case értéke
          end,
    %% Második case-ben a jobb fa rendezettségét vizsgáljuk, és azt, hogy a
    %% teljes fa rendezett-e (tudjuk, hogy {C,Bfa,level} fa rendezett).
    if CBR  -> case mmrendezett(Jfa) of
                   {JMin,JMax,JR} -> {Min,JMax,JR andalso C<JMin}; % eredmény
                   JR             -> {Min,C,JR}   % eredmény
               end;
       true -> false   % eredmény (CBR=:=false, tehát nem rendezett)
    end.

```