```
%                    Deklaratív Programozás gyakorlat
%                   Prolog programozás: listák, gráfok
%                                MEGOLDÁSOK
% ---------------------------------------------------------------


% 1.  Beszúrás rendezett listába

%     % insert_ord(+RL0, +Elem, RL): Az RL monoton növő számlista úgy áll
%     % elő, hogy az RL0 szigorúan növő számlistába beszúrjuk az Elem számot,
%     % feltéve hogy Elem nem eleme az RL0 listának; egyébként RL = RL0.

%     | ?- insert_ord([1,3,5,8], 6, L).
%     L = [1,3,5,6,8] ? ;
%     no
%     | ?- insert_ord([1,3,5,8], 3, L).
%     L = [1,3,5,8] ? ;
%     no

%     Használjon feltételes szerkezetet!

% insert_ord(+RL0, +Elem, ?RL): Az RL monoton növő számlista úgy áll
% elő, hogy az RL0 szigorúan növő számlistába beszúrjuk az Elem számot,
% feltéve hogy Elem nem eleme az RL0 listának; egyébként RL = RL0.
insert_ord([], E, [E]).
insert_ord(L0, E, L) :-
        L0 = [X|L1],
        (   X > E -> L = [E|L0]
        ;   X =:= E -> L = L0
        ;   L = [X|L2],
            insert_ord(L1, E, L2)
        ).

% insert_ord_1(+RL0, +Elem, ?RL): ugyanaz mint
%   insert_ord(+RL0, +Elem, ?RL), de feltételes szerkezet használata
% nélkül. Kevésbé hatékony, mert választási pontot hagy maga után.
insert_ord_1([], E, [E]).
insert_ord_1([E|L0], E, [E|L0]).
insert_ord_1([X|L0], E, [E,X|L0]) :-
        X > E.
insert_ord_1([X|L0], E, [X|L]) :-
        X < E,
        insert_ord_1(L0, E, L).


% insert_ord_2(+RL0, +Elem, ?RL): ugyanaz mint
%   insert_ord(+RL0, +Elem, ?RL), de vágóval
% Kicsit egyszerűbb mint az előző két változat, mert csak
% háromfelé ágazik (insert_ord_1 1. és 3. klózát itt most
% egyetlen klóz, a 3. helyettesíti)
insert_ord_2([X|L0], E, L) :-
        X < E, !, L = [X|L1], insert_ord_2(L0, E, L1).
insert_ord_2([E|L0], E, L) :-
        !, L = [E|L0].
insert_ord_2(L0, E, [E|L0]).


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

% A 'graph' adatstruktúrát a következő Mercury-szerű típusdefiníciókkal
% definiáljuk:

% % :- type graph == list(edge).
% % :- type edge ---> node-node.
% % :- type node == atom.
```

```
% Eszerint egy Prolog kifejezés a 'graph' típusba tartozik, ha X-Y alakú
% struktúrák listája, ahol X és Y névkonstansok (atomok).

% Az [a1-b1,a2-b2,...,an-bn] 'graph' típusú kifejezés azt az irányítatlan
% gráfot írja le, amelynek csomópontjai  a1,..,an,b1,...,bn, és
% egy (irányítatlan) él vezet ai és bi között, minden i=1,...,n esetén.
% (Megjegyzés: az így megadott gráfoknak nyilván nem lehet izolált pontja.)

% Például az  [a-b,a-c], [a-c,b-a], [b-a,a-c], [c-a,a-b] stb.  mind
% ugyanazt a (matematikai értelemben vett) gráfot írják le.

% Az [a-b,a-c,b-c,b-d,b-e,c-d,c-e,d-e] kifejezés az alábbi gráfot írja le:

%               a
%              / \
%             /   \
%            /     \
%           b-------c
%           |\     /|
%           |  \ /  |
%           |  / \  |
%           |/     \|
%           d-------e

% Gyerekkorukban találkozhattak azzal a feladattal, hogy ezt a gráfot egy
% folytonos vonallal rajzolják meg.

% Egy 'graph' típusú [a1-b1,a2-b2,...,an-bn] Prolog listát folytonos
% vonalnak hívunk, ha b1=a2, b2=a3, ...,    b(n-1) = an.


% 2.  Írja meg az alábbi fejkommentnek  megfelelő  draw/2 Prolog eljárást

%     % draw(+G, -L): Az L folytonos vonal "megrajzolja" a  G gráfot, azaz az
%     % L folytonos vonal ugyanazt a matematikai értelemben vett gráfot írja
%     % le, mint a G Prolog kifejezés.

%     Tehát a "| ?- draw(G, L)." hívás, ahol G adott és L egy változó,
%     felsorolja L-ben az összes olyan folytonos vonalat, amely "megrajzolja"
%     G-t.

%     Törekedjék minél egyszerűbb megoldásra, nem kell a hatékonysággal
%     foglalkoznia. Használhatja a lists könyvtár eljárásait.

%     | ?- draw([a-b,a-c], L).
%     L = [b-a,a-c] ? ;
%     L = [c-a,a-b] ? ;
%     no
%     | ?- draw([a-b,a-c,b-c,b-d,b-e,c-d,c-e,d-e], L), L = [d-e|_].
%     L = [d-e,e-b,b-a,a-c,c-b,b-d,d-c,c-e] ? ;
%     L = [d-e,e-b,b-a,a-c,c-d,d-b,b-c,c-e] ? ;
%     L = [d-e,e-b,b-c,c-a,a-b,b-d,d-c,c-e] ? ;
%     L = [d-e,e-b,b-c,c-d,d-b,b-a,a-c,c-e] ? ;
%     L = [d-e,e-b,b-d,d-c,c-a,a-b,b-c,c-e] ? ;
%     L = [d-e,e-b,b-d,d-c,c-b,b-a,a-c,c-e] ? ;
%     L = [d-e,e-c,c-a,a-b,b-c,c-d,d-b,b-e] ? ;
%     L = [d-e,e-c,c-a,a-b,b-d,d-c,c-b,b-e] ? ;
%     L = [d-e,e-c,c-b,b-a,a-c,c-d,d-b,b-e] ? ;
%     L = [d-e,e-c,c-b,b-d,d-c,c-a,a-b,b-e] ? ;
%     L = [d-e,e-c,c-d,d-b,b-a,a-c,c-b,b-e] ? ;
%     L = [d-e,e-c,c-d,d-b,b-c,c-a,a-b,b-e] ? ;

%     no
```

```
:- use_module(library(lists), [select/3]).

% draw(+G, -L): Az L folytonos vonal "megrajzolja" a  G gráfot, azaz az
% L folytonos vonal ugyanazt a matematikai értelemben vett gráfot írja
% le, mint a G Prolog kifejezés.
draw(G, L) :-
        draw(G, _, L).

% draw(+G,  ?KP, -L):  Az L  folytonos vonal  a KP  kezdőpontból  indul és
% "megrajzolja"  a  G  gráfot,  azaz   az  L  folytonos  vonal  ugyanazt  a
% matematikai értelemben vett gráfot írja le, mint a G Prolog kifejezés.
draw([], _, []).
draw(G, P, [P-Q|L]) :-
        select_edge(P, Q, G, G1),
        draw(G1, Q, L).

% select_edge(P, Q, G, G1):
% A G irányítatlan gráfból elhagyható a P-Q él és marad a G1 gráf.
select_edge(P, Q, G, G1) :-
        select(E, G, G1),
        (   E = P-Q
        ;   E = Q-P
        ).

% draw1(+G, -L): ugyanaz, mint draw/2, de csak egy
% segédeljárás használatával (select_edge/4)
draw1([], []).
draw1(G, [P-Q|L]) :-
        select_edge(P, Q, G, G1),
        (   G1 = [] -> L = []
        ;   L = [Q-_|_],
            draw1(G1, L)
        ).

% draw2(+G, -L): ugyanaz, mint draw/2, csak segédeljárás nélkül.
draw2([], []).
draw2(G, [A-B|L]) :-
        select(E, G, G1),
        (   E = A-B
        ;   E = B-A
        ),
        (   G1 = [] -> L = []
        ;   L = [B-_|_],
            draw2(G1, L)
        ).
```

```
% 3.  (Otthoni feladat)

%     Írjon Prolog eljárást egy gráf fokszámlistájának előállítására. A
%     fokszámlista típusa:

%     % :- type degrees == list(node_degree).
%     % :- type node_degree --> node - degree.
%     % :- type degree == int.

%     A fokszámlista tehát egy olyan lista, amelyenek elemei Cs-N alakú
%     párok, ahol Cs a gráf egy csomópontja, és N a Cs csomópont
%     fokszáma. A csomópontok tetszőleges sorrendben szerepelhetnek a
%     fokszámlistában.

%     % degree_list(G, Ds): A G gráf fokszámlistája Ds.

%     | ?- degree_list([b-a,a-c], Ds).
%     Ds = [b-1,a-2,c-1] ? ; no

% degree_list(G, Ds): A G gráf fokszámlistája Ds.
degree_list([], []).
degree_list([A-B|G], Ds) :-
        degree_list(G, Ds0),
        incr_node_degree(Ds0, A, Ds1),
        incr_node_degree(Ds1, B, Ds).

% incr_node_degree(Ds0, A, Ds): A Ds0 fokszámlistából A fokszámának eggyel
% való növelésével áll elő a Ds fokszámlista.
incr_node_degree([], A, [A-1]).
incr_node_degree([N-D|Ds0], A, Ds) :-
        (   A = N -> D1 is D+1, Ds = [A-D1|Ds0]
        ;   Ds = [N-D|Ds1],
            incr_node_degree(Ds0, A, Ds1)
        ).

% degree_list2(G, Ds): A G gráf fokszámlistája Ds. (Jobbrekurzív változat)
degree_list2(G, Ds) :-
        degree_list2(G, [], Ds).

% degree_list2(G, Ds0, Ds): A G gráf fokszámlistáját Ds0 elé
% fűzve kapjuk Ds-t.
degree_list2([], Ds0, Ds0).
degree_list2([A-B|G], Ds0, Ds) :-
        incr_node_degree(Ds0, A, Ds1),
        incr_node_degree(Ds1, B, Ds2),
        degree_list2(G, Ds2, Ds).
```

```
% 4.  (Otthoni feladat)

%     Írjon idraw/2 néven egy Prolog eljárást, amelynek jelentése azonos a
%     2. feladatban szereplő draw/2 eljárással! Törekedjék minél hatékonyabb
%     megoldásra! Használhatja az ugraphs könyvtárat.

:- use_module(library(ugraphs)).

idraw(G, L) :-
        vertices_edges_to_ugraph([], G, Graph),
        symmetric_closure(Graph, SGraph),
        reduce(SGraph, [_]),    % összefüggő a gráf
        degree_list2(G, Ds0),
        (   select(A-D1, Ds0, Ds1), D1 mod 2 =:= 1,
            select(B-D2, Ds1, Ds2), D2 mod 2 =:= 1 ->
            \+ ( member(_C-D3, Ds2), D3 mod 2 =:= 1 ),
            (   L = [A-_|_]
            ;   L = [B-_|_]
            )
        ;   true
        ),
        draw(G, L).



big_house([a-b,a-c,b-c,b-d,b-e,c-d,c-e,d-e,
           a-f,f-b,f-c,g-f,g-b,g-c,h-g,h-b,h-c,
           i-h,i-b,i-c,j-i,j-b,j-c,k-j,k-b,k-c]).

%               a  ------ f ------ g ------ h ------ i ------ j ------ k
%              / \       / \       / \       / \       / \       / \
%             /   \     +   +     +   +     +   +     +   +     +   +
%            /     \   b     c   b     c   b     c   b     c   b     c
%           b-------c
%           |\     /|
%           |  \ /  |
%           |  / \  |
%           |/     \|
%           d-------e

:- use_module(library(lists), [prefix_length/3]).
:- use_module(library(between), [between/3]).


test(Pred) :-
    big_house(G0),
    length(G0, Len),
    N is (Len-8)//3,
    (   between(1, N, I),
        NoOfEdges is 8+3*I,
        prefix_length(G0, G, NoOfEdges),
        Goal =.. [Pred,G,_],
        statistics(runtime, _),
        \+ Goal,            % Goal is bound to fail
        statistics(runtime, [_,T]),
        format('Pred:~|~t~w~6+, edges:~|~t~w~4+, time to fail:~|~t~3d~8+ sec\n',
               [Pred, NoOfEdges, T]),
        fail
    ;   true
    ).


test :-
    test(idraw), nl, test(draw).
```

```
/*
| ?- test.
Pred: idraw, edges:  11, time to fail:   0.000 sec
Pred: idraw, edges:  14, time to fail:   0.000 sec
Pred: idraw, edges:  17, time to fail:   0.000 sec
Pred: idraw, edges:  20, time to fail:   0.000 sec
Pred: idraw, edges:  23, time to fail:   0.000 sec
Pred: idraw, edges:  26, time to fail:   0.000 sec
Pred:  draw, edges:  11, time to fail:   0.000 sec
Pred:  draw, edges:  14, time to fail:   0.050 sec
Pred:  draw, edges:  17, time to fail:   1.120 sec
Pred:  draw, edges:  20, time to fail:  26.670 sec
Pred:  draw, edges:  23, time to fail: 699.370 sec
  C-c C-cProlog interruption (h for help)? a
% Execution aborted
| ?-
*/



% 5.  Írjon Prolog eljárást amely a bemenetként kezdődő listáról eldönti,
%     hogy egy platóval kezdődik-e, és ha igen, visszaadja a maximális plató
%     hosszát és az ezutáni (maradék) elemek listáját.
%
%     % pl_kezdetu(L, Len, M): Az atomokból álló L lista egy Len hosszúságú
%     % maximális platóval kezdődik, amelyet az M maradéklista követ.
%
%     | ?- pl_kezdetu([a,b,a,c,c,c,b,b], Len, M).
%     no
%     | ?- pl_kezdetu([c,c,c,b,b], Len, M).
%     Len = 3, M = [b,b] ? ; no
%     | ?- pl_kezdetu([b,b], Len, M).
%     Len = 2, M = [] ? ; no
%     | ?- pl_kezdetu([b], Len, M).
%     no
%     | ?- pl_kezdetu([], Len, M).
%     no
%     | ?-

% maxazonos(L0, M, A, N0, N): Az L0 lista elejerol leszedheto k db A es
% marad egy nem A-val kezdodo M, tovabba N=N0+k.
maxazonos(L0, M, A, N0, N) :-
    (   L0 = [A|L1] ->
        N1 is N0+1,
         maxazonos(L1, M, A, N1, N)
    ;   M = L0, N = N0
    ).

pl_kezdetu([A,A|L], Len, M) :-
    maxazonos(L, M, A, 2, Len).

% Kevésbé hatékony, de segédeljárás nélküli.
pl_kezdetu1([A,A], 2, []).
pl_kezdetu1([A|L], Len, M) :-
        L = [A|L1],
        L1 = [B|_],
        (  B = A -> pl_kezdetu1(L, Len1, M), Len is Len1+1
        ;  Len = 2, M = L1
        ).
```

```
% 6. Írjon olyan Prolog eljárást, amely felsorolja atomok egy adott
%    listájában található maximális platókat, megadva a plató hosszát és az
%    ismétlődő elemet.
%
%    % plato(+L, ?Len, ?X): Az L listában található egy Len hosszúságú,
%    % X elemekből képzett maximális plató.
%
%    | ?- plato([a,b,b,b,b,a,a,c,b,b], Len, X).
%    Len = 4, X = b ? ;
%    Len = 2, X = a ? ;
%    Len = 2, X = b ? ;
%    no

plato(L, Len, X) :-
    L = [X0|L1],
    (   pl_kezdetu(L, Len0, M) ->
            (   X = X0, Len = Len0
            ;   plato(M, Len, X)
            )
    ;   plato(L1, Len, X)
    ).

%   % elso_plato(L, Len, X, M): Az L listában van plató, Az első maximális
%   % plató Len hosszúságú és X elemekből képzett, továbbá ezt a platót egy
%   % M maradéklista követi.
%
%   | ?- elso_plato([a,b,b,b,b,a,a,c,b,b], Len, X, M).
%   Len = 4, X = b, M = [a,a,c,b,b] ? ; no

elso_plato(L, Len, X, M) :-
        (   pl_kezdetu(L, Len, M) ->
            L = [X|_]
        ;   L = [_|L1],
            elso_plato(L1, Len, X, M)
        ).

% Nem hatékony, mert kétszer hívja elso_plato/4-et
plato1(L, Len, X) :-
        elso_plato(L, Len, X, _M).
plato1(L, Len, X) :-
        elso_plato(L, _Len, _X, M),
        plato1(M, Len, X).

% plato1 ekvivalens átírása olyan formába, hogy a két klóz pontosan
% ugyanazzal az elso_plato hívással kezdődjék:
plato1a(L, Len, X) :-
        elso_plato(L, Len0, X0, M0),
        Len = Len0, X = X0.
plato1a(L, Len, X) :-
        elso_plato(L, Len0, X0, M0),
        plato1a(M0, Len, X).

% A fenti kód futóképes, de az alábbi figyelmeztetéseket adja.
% * [M0] - singleton variables
% * Approximate lines: 336-339, file: '.../dp18a-gy4.pl'
% * [X0,Len0] - singleton variables
% * Approximate lines: 339-342, file: '.../dp18a-gy4.pl'

% Ezeket célszerű kiküszöbölni, pl. egy aláhúzást rakva az érintett
% változók elé

% plato1a logikailag ekvivalens, de lényegesen hatékonyabb átírása
% diszjukcióvá:

plato2(L, Len, X) :-
        elso_plato(L, Len0, X0, M0),
        (   Len = Len0, X = X0
        ;   plato1a(M0, Len, X)
        ).
```

```
:- use_module(library(lists),[last/2]).

plato3(L, Len, X) :-        % Az L listában található egy Len hosszúságú,
                            % X elemekből képzett maximális plató         HA
    L2 = [X,X|_],           % L2 két azonos X elemmel kezdődő lista       ÉS
    append(L1, L23, L),     % L szétszedhető L1 és L23 részlistákra       ÉS
    append(L2, L3, L23),    % L23 szétszedhető L2 és L3 részlistákra      ÉS
    minden_eleme(L2, X),    % Az L2 lista minden eleme X                  ÉS
    \+ L3 = [X|_],          % L3 nem X-szel kezdődő lista  (1)            ÉS
    \+ last(L1, X),         % L1 nem X-szel végződő lista  (2)            ÉS
    length(L2, Len).        % L2 hossza Len.

% Vegyük észre, hogy az (1) ill. (2) feltételek akkor is teljesülnek, ha
% az L3 ill. L1 listák üresek!

% minden_eleme(L, X): Az L lista minden eleme X-szel egyenlő.
minden_eleme([], _).
minden_eleme([X|L], X) :-
        minden_eleme(L, X).


plato4(L, Len, X) :-        % Az L listában található egy Len hosszúságú,
                            % X elemekből képzett maximális plató         HA
    L2 = [X,X|_],           % L2 két azonos X elemmel kezdődő lista       ÉS
    append(L1, L23, L),     % L szétszedhető L1 és L23 részlistákra       ÉS
    append(L2, L3, L23),    % L23 szétszedhető L2 és L3 részlistákra      ÉS
    \+ (                    % nem teljesül, hogy van olyan Y, amelyre
         member(Y, L2),     %      ( Y eleme L2-nek ÉS
         Y \= X             %        Y nem egyenlő X-szel
       ),                   %      )                                      ÉS
    \+ L3 = [X|_],          % L3 nem X-szel kezdődő lista  (1)            ÉS
    \+ last(L1, X),         % L1 nem X-szel végződő lista  (2)            ÉS
    length(L2, Len).        % L2 hossza Len.


%
% 7. (Otthoni feladat)
%
%    Az előző feladat kiterjesztéseként írjon olyan Prolog eljárást, amely
%    felsorolja atomok egy adott listájában található maximális platókat,
%    megadva a plató kezdőindexét (1-től számozva), hosszát és az ismétlődő
%    elemet.
%
%    % plato(L, I, Len, X): Az L listában az I-edik elemtől kezdődően
%    % egy X elemekből képzett, Len hosszúságú maximális plató található.
%
%    | ?- plato([a,b,b,b,b,a,a,c,b,b], I, Len, X).
%    I = 2, Len = 4, X = b ? ;
%    I = 6, Len = 2, X = a ? ;
%    I = 9, Len = 2, X = b ? ;
%    no


plato(L, I, Len, X) :-
    plato(L, 1, I, Len, X).


plato(L, I0, I, Len, X) :-
    L = [X0|L1],
    (   pl_kezdetu(L, Len0, M) ->
            (   X = X0, Len = Len0, I = I0
            ;   I1 is I0+Len0, plato(M, I1, I, Len, X)
            )
    ;   I1 is I0+1, plato(L1, I1, I, Len, X)
    ).
```