```
%                     Deklaratív programozás gyakorlat
%     Prolog programozás: egyszerű listakezelés, bináris fák, aritmetika
%                                MEGOLDÁSOK
% ---------------------------------------------------------------------

% "P". Témakör: programok írása
% =============================

% 1.  Számsorozat generálása

%     % seq(+N, +M, -L): Az L lista M-N+1 hosszú, elemei 1 különbségű
%     % számtani sorozatot alkotnak, és L első eleme (ha van) N,
%     % ahol N és M egész számok.

%     | ?- seq(2, 4, L).
%     L = [2,3,4] ? ; no
%     | ?- seq(4, 2, L).
%     no
%     | ?- seq(4, 3, L).
%     L = [] ? ; no
%     | ?- seq(-4, -2, L).
%     L = [-4,-3,-2] ? ; no

% seq(+N, +M, -L): Az L lista M-N+1 hosszú, elemei 1 különbségű számtani
% sorozatot alkotnak, és L első eleme (ha van) N, ahol N és M egész számok.
seq(N, M, []) :-
        M =:= N - 1.
seq(N, M, [N|Seq]) :-
        M >= N,
        N1 is N+1,
        seq(N1, M, Seq).

% P2. Számintervallum felsorolása

%     % max(+N, ?X): X egy egész szám, melyre 0 < X =< N, ahol N adott
%     % egész szám. Az eljárás a fenti feltételeknek megfelelő X számokat
%     % sorolja fel. A felsorolás sorrendjére nem teszünk megkötést.

%     | ?- max(1,X).
%     X = 1 ? ; no
%     | ?- max(4,X).
%     X = 4 ? ; X = 3 ? ; X = 2 ? ; X = 1 ? ; no
%     | ?- max(4,3).
%     yes
%     | ?- max(4,5).
%     no
%     | ?- max(0,X).
%     no
%     | ?- max(-6,X).
%     no

% max(+N, ?X): X egy egész szám, melyre 0 < X =< N.
max(N, N) :-
        N > 0.
max(N, X) :-
        N > 1,
        N1 is N-1,
        max(N1, X).
```

```
% 3.  Hatványozás

%     % hatv(+A, +E, -H):  H = A ^ E, ahol A egész szám, E >= 0 egész szám.

%     | ?- hatv(3, 5, X).
%     X = 243 ? ; no

% hatv(+A, +E, -H):  H = A ^ E, ahol A egész szám, E >= 0 egész szám.
hatv(_A, 0, 1).
hatv(A, N, H) :-
        N > 0,
        N1 is N-1,
        hatv(A, N1, H1),
        H is A*H1.

% 4.  Fa csomópontjainak megszámolása

%     Egy fa csomópontjainak száma a benne előforduló node/2 struktúrák
%     száma.

%     % fa_pontszama(*Fa, -N): A Fa bináris fa csomópontjainak száma N.

%     | ?- fa_pontszama(node(leaf(1),node(leaf(2),leaf(3))), N).
%     N = 2 ? ; no
%     | ?- fa_pontszama(node(leaf(1),node(leaf(2),node(leaf(4),leaf(3)))),N).
%     N = 3 ? ; no

% fa_pontszama(+Fa, -N): A Fa bináris fa csomópontjainak száma N.
fa_pontszama(leaf(_), 0).
fa_pontszama(node(L,R), P) :-
        fa_pontszama(L, LP),
        fa_pontszama(R, RP),
        P is LP+RP+1.

% 5.  Fa minden levélértékének növelése

%     % fa_noveltje(*Fa0, ?Fa): Fa úgy áll elő a Fa0 bináris fából, hogy az
%     % utóbbi minden egyes levelében levő értéket 1-gyel megnöveljük.

%     | ?- fa_noveltje(node(leaf(1),node(leaf(2),leaf(3))), Fa).
%     Fa = node(leaf(2),node(leaf(3),leaf(4))) ? ; no

% fa_noveltje(+Fa0, ?Fa): Fa úgy áll elő a Fa0 bináris fából, hogy az
% utóbbi minden egyes levelében levő értéket 1-gyel megnöveljük.
fa_noveltje(leaf(X), leaf(Y)) :-
        Y is X+1.
fa_noveltje(node(L,R), node(NL,NR)) :-
        fa_noveltje(L, NL),
        fa_noveltje(R, NR).

% 6.  Lista hosszának meghatározása

%     Egy lista hosszának az elemei számát nevezzük.

%     % lista_hossza(*Lista, -Hossz): A Lista egészlista hossza Hossz.

%     | ?- lista_hossza([1,3,5], H).
%     H = 3 ? ; no

% lista_hossza(+Lista, -Hossz): A Lista egészlista hossza Hossz.
lista_hossza([], 0).
lista_hossza([_|L], H) :-
        lista_hossza(L, H0),
        H is H0+1.
```

```
% 6*. (szorgalmi, otthoni feladat) Lista hosszának meghatározása --
%     jobbrekurzív változat

%     % lista_hossza2(*Lista, -Hossz): A Lista egészlista hossza Hossz.
%     % Jobbrekurzív változat
%     Segédeljárás szükséges.

% lista_hossza2(+Lista, +H0, H): A Lista egészlista hossza H-H0.
lista_hossza2([], H0, H0).
lista_hossza2([_|L], H0, H) :-
        H1 is H0+1,
        lista_hossza2(L, H1, H).

% lista_hossza2(+Lista, -Hossz): A Lista egészlista hossza Hossz.
lista_hossza2(Lista, Hossz) :-
        lista_hossza2(Lista, 0, Hossz).

% 7.  Egészlista minden elemének növelése

%     % lista_noveltje(*L0, ?L): Az L egészlista úgy áll elő az L0
%     % egészlistából, hogy az utóbbi minden egyes elemét 1-gyel megnöveljük.

%     | ?- lista_noveltje([1,5,2], L).
%     L = [2,6,3] ? ; no

% lista_noveltje(+L0, ?L): Az L számlista úgy áll elő az L0
% számlistából, hogy az utóbbi minden egyes elemét 1-gyel megnöveljük.
lista_noveltje([], []).
lista_noveltje([X|L], [NX|NL]) :-
        NX is X+1,
        lista_noveltje(L, NL).

% 8.  Egy lista utolsó elemének meghatározása

%     % lista_utolso_eleme(*L, ?Ertek): Az L egészlista utolsó eleme Ertek.

%     | ?- lista_utolso_eleme([5,1,2,8,7], E).
%     E = 7 ? ; no

% lista_utolso_eleme(+L, ?Ertek): Az L egészlista utolsó eleme Ertek.
lista_utolso_eleme([E], E).
lista_utolso_eleme([_|L], E) :-
        lista_utolso_eleme(L, E).

% 9.  Egy fa leveleiben található értékek felsorolása

%     % fa_levelerteke(*Fa, -Ertek): A Fa bináris fa egy levelében található
%     % érték az Ertek.

%     Az eljárás nemdeterminisztikus módon sorolja fel az összes
%     levélértéket.  A felsorolás sorrendjére nem teszünk megkötést.

%     | ?- fa_levelerteke(node(leaf(1),node(leaf(2),leaf(3))), E).
%     E = 1 ? ; E = 2 ? ; E = 3 ? ; no

% fa_levelerteke(*Fa, -Ertek): A Fa bináris fa egy levelében található
% érték az Ertek.
fa_levelerteke(leaf(E), E).
fa_levelerteke(node(L,_), E) :-
        fa_levelerteke(L, E).
fa_levelerteke(node(_,R), E) :-
        fa_levelerteke(R, E).
```

```
% 10. Egy fa részfáinak a felsorolása

%     Egy fa (nem feltétlenül valódi) részfájának nevezzük saját magát,
%     valamint - ha a fa egy csomópont - akkor a bal és jobboldali ág
%     részfáit.

%     % fa_reszfaja(*Fa, -Resz): Resz a Fa bináris fa részfája.

%     A fenti eljárás nemdeterminisztikus, azaz többféleképpen sikerül:
%     a Resz változóban fel kell sorolnia a Fa összes részfáját. A felsorolás
%     sorrendjére nem teszünk megkötést.

%     | ?- fa_reszfaja(node(leaf(1),node(leaf(2),leaf(3))), Fa).
%     Fa = node(leaf(1),node(leaf(2),leaf(3))) ? ;
%     Fa = leaf(1) ? ;
%     Fa = node(leaf(2),leaf(3)) ? ;
%     Fa = leaf(2) ? ;
%     Fa = leaf(3) ? ; no

%     Gondolja meg, hogy a predikátum klózai sorrendjének változtatásakor
%     hogyan változik a felsorolás sorrendje!

% fa_reszfaja(+Fa, -Resz): Resz a Fa bináris fa részfája.
fa_reszfaja(Fa, Fa).
fa_reszfaja(node(L,_), Fa) :-
        fa_reszfaja(L, Fa).
fa_reszfaja(node(_,R), Fa) :-
        fa_reszfaja(R, Fa).

%     A fa_reszfaja eljárás felhasználásával írja meg a 9. feladat
%     megoldását, fa_levelerteke2 néven!

% fa_levelerteke2(+Fa, -Ertek): A Fa bináris fa egy levelében található
% érték az Ertek.
fa_levelerteke2(Fa, E) :-
        fa_reszfaja(Fa, leaf(E)).


% 11. Egy lista prefixumainak a felsorolása

%     Egy L n-elemű lista prefixumának nevezzünk egy listát, ha az az L első
%     k elemét tartalmazza (az L-beli sorrendben), ahol 0 =< k =< n.

%     % lista_prefixuma(*L0, -L): L az L0 egészlista prefixuma.

%     A fenti eljárás nemdeterminisztikus, azaz többféleképpen sikerül: az L
%     változóban fel kell sorolnia a L0 összes prefixumát. A felsorolás
%     sorrendjére nem teszünk megkötést.

%     | ?- lista_prefixuma([1,4,2], Sz).
%     Sz = [1,4,2] ? ;
%     Sz = [1,4] ? ;
%     Sz = [1] ? ;
%     Sz = [] ? ; no

%     Gondolja meg, hogy a predikátum klózai sorrendjének változtatásakor
%     hogyan változik a felsorolás sorrendje!

% lista_prefixuma(*L0, -L): L az L0 egészlista prefixuma.
lista_prefixuma([X|L], [X|P]) :-
        lista_prefixuma(L, P).
lista_prefixuma(_, []).
```