```
%
%                    Deklaratív Programozás gyakorlat
%                Prolog programozás: meta-logikai eljárások

%

% 1.  Atomok szeletelése

%     Egy A atom prefixumának nevezünk egy P atomot, ha P az A első
%     valahány karakterét tartalmazza, az A-beli sorrend megtartásával.

%     % atom_prefix(+Atom, ?Prefix, +N): Atom-nak Prefix N hosszú prefixuma.
%     % Másszóval: az Atom első N karakteréből képzett névkonstans a Prefix atom.

%     | ?- atom_prefix(abcde, Prefix, 0).
%     Prefix = '' ? ; no
%     | ?- atom_prefix(abcde, Prefix, 3).
%     Prefix = abc ? ; no
%     | ?- atom_prefix(abcde, Prefix, 5).
%     Prefix = abcde ? ; no
%     | ?- atom_prefix(abcde, Prefix, 6).
%     no

%     Nem használhatja a sub_atom/5 beépített eljárást!
%     Ötlet: használja az atom_codes és prefix_length eljárásokat.

:- use_module(library(lists)).

atom_prefix(Atom, Prefix, N) :-
        atom_codes(Atom, Codes),
        length(PrefixCodes, N),                 % (1)
        append(PrefixCodes, _, Codes),          % (2)
        atom_codes(Prefix, PrefixCodes).


% Ha az (1) és (2) sorokat felcseréljük, a kód helyes marad, de
% sokkal lassabb lesz. Miért?


% 2.  Általános Prolog kifejezés bizonyos részkifejezéseinek felsorolása
%
%     % reszatom(+K, ?A): A a K általános Prolog kifejezésben
%     % előforduló atom.
%
%     | ?- reszatom(a, X).
%     X = a ? ;
%     no
%     | ?- reszatom(f(X,[1,3,b],g(2,1,a0)), A).
%     A = b ? ;
%     A = [] ? ;
%     A = a0 ? ;
%     no
%
%     Megjegyzés: a struktúranevet nem tekintjük a struktúrakifejezés
%     részének.

reszatom0(K, A) :-
        (   atom(K) ->
            K = A
        ;   compound(K) ->
            K =.. [_Fun|Args],
            lista_reszatom0(Args, A)
        ).

% Lista_reszatom(+L, ?A): A az L lista egy elemében előforduló atom.
lista_reszatom0(Args, A) :-
        member(Arg, Args),
        reszatom0(Arg, A).
```

```
reszatom_rossz(K, A) :-
        (   atom(K) ->
            K = A
        ;   compound(K) ->
            K =.. [_Fun|Args],
            reszatom_rossz(Args, A)
        ).

% | ?- trace, reszatom_rossz(f(a), A).
% % The debugger will first creep -- showing everything (trace)
%        1      1 Call: reszatom_rossz(f(a),_933) ?
%        2      2 Call: reszatom_rossz([a],_933) ?
%        3      3 Call: reszatom_rossz([a,[]],_933) ?
%        4      4 Call: reszatom_rossz([a,[[]]],_933) ?
%        5      5 Call: reszatom_rossz([a,[[[]]]],_933) ?
%        6      6 Call: reszatom_rossz([a,[[[[]]]]],_933) ?
%        7      7 Call: reszatom_rossz([a,[[[[[]]]]]],_933) ?
%        8      8 Call: reszatom_rossz([a,[[[[[[]]]]]],_933) ?
%        9      9 Call: reszatom_rossz([a,[[[[[[[[]]]]]]]],_933) ?
%       10     10 Call: reszatom_rossz([a,[[[[[[[[]]]]]]]],_933) ?
%       11     11 Call: reszatom_rossz([a,[[[[[[[[...]]]]]]]],_933) ?
%       12     12 Call: reszatom_rossz([a,[[[[[[[[...]]]]]]]],_933) ?
%       13     13 Call: reszatom_rossz([a,[[[[[[[[...]]]]]]]],_933) ?


% A reszatom0/2 eljárásból, a lista_reszatom0/2 nem-rekurzív eljárás
% kiküszöbölhető, így jutunk az alábbi reszatom/2 változathoz:
reszatom(K, A) :-
        (   atom(K) ->
            K = A
        ;   compound(K) ->
            K =.. [_Fun|Args],
            member(Arg, Args),
            reszatom(Arg, A)
        ).




% 3.  Általános Prolog kifejezés bizonyos részkifejezéseinek akkumulálása
%
%     % osszege(+K, ?Ossz): Ossz a K kifejezésben előforduló egész számok
%     % összege.
%
%     | ?- osszege(a, S).
%     S = 0 ? ;
%     no
%     | ?- osszege(1, S).
%     S = 1 ? ;
%     no
%     | ?- osszege(f(X,[1,3,b],g(2,1,a0)), S).
%     S = 7 ? ;
%     no

osszege(K, Sum) :-
        osszege(K, 0, Sum).

% a K kifejezésben előforduló egész számok összege + Sum0 = Sum.
osszege(K, Sum0, Sum) :-
        (   integer(K) ->
            Sum is Sum0+K
        ;   compound(K) ->
            K =.. [_Fun|Args],
            osszege_lista(Args, Sum0, Sum)
        ;   Sum = Sum0
        ).

% osszege_lista(+KL, +Ossz0, ?Ossz): A KL listában előforduló egész számok
% összege plusz az Ossz0 szám egyenlő az Ossz számmal.
```

```
osszege_lista([], Sum, Sum).
osszege_lista([X0|L0], Sum0, Sum) :-
        osszege(X0, Sum0, Sum1),
        osszege_lista(L0, Sum1, Sum).



osszege_rossz(K, Sum0, Sum) :-
        (   integer(K) ->
            Sum is Sum0+K
        ;   compound(K) ->
            K =.. [_Fun|Args],
            osszege_rossz(Args, Sum0, Sum)
        ;   Sum = Sum0
        ).

osszege_rossz2(K, Sum0, Sum) :-
        (   integer(K) ->
            Sum is Sum0+K
        ;   compound(K) ->
            K =.. [_Fun, A|Args],
            osszege_rossz2(A, Sum0, Sum1),
            osszege_rossz2(Args, Sum1, Sum)
        ;   Sum = Sum0
        ).


% 5.  M1 mintafeladat segédeljárása:

%     Írjon Prolog nyelven egy olyan eljárást, amely előállítja egy konstans
%     értékét egy helyettesítési lista alapján. A helyettesítési lista minden
%     eleme Név-Szám alakú, ahol Szám a Név névkonstans helyettesítési értéke. Egy
%     számkonstans helyettesítési értéke önmaga, egy a helyettesítési listában nem
%     szereplő atom helyettesítési értéke pedig 0.  Ha egy névkonstans többször
%     szerepel a helyettesítési listában, akkor az első előfordulást szabad csak
%     figyelembe venni.

% helyettesitese(+K, +HL, ?E): A K konstansnak a HL behelyettesítési
% lista szerinti értéke E.
helyettesitese(K, HL, E) :-
        (   number(K) -> E = K
        ;   atom(K),
            (   memberchk(K-E1, HL) -> E = E1
            ;   E = 0
            )
        ).

helyettesitese2(K, _, K) :-
        number(K).
helyettesitese2(K, [M-Sz|HL], E) :-
        atom(K),
        (   K = M -> E = Sz
        ;   helyettesitese2(K, HL, E)
        ).
helyettesitese2(K, [], 0) :-
        atom(K).


% 6.  M1 teljes feladat:

%     A helyettesitese/3 eljárás segítségével írjon olyan Prolog eljárást,
%     amely egy többváltozós kifejezés adott lista szerinti behelyettesítési
%     értékét számítja ki! A kifejezést egy olyan Prolog adatstruktúrával
%     adjuk meg, amely atomokból és számokból az 'is' beépített eljárás által
%     megengedett egy- ill. kétargumentumú műveletekkel épül fel.

% erteke(+Kif, +Hely, ?Ert): A Kif kifejezés értéke a Hely behelyettesítési
% lista által adott helyen Ert.
erteke(K, HL, E) :-
        atomic(K), helyettesitese(K, HL, E).
```

```
erteke(K, HL, E) :-
        K =.. [Op,A1],
        erteke(A1, HL, A1E),
        EKif =.. [Op,A1E],
        E is EKif.
erteke(K, HL, E) :-
        K =.. [Op,A1,A2],
        erteke(A1, HL, A1E),
        erteke(A2, HL, A2E),
        EKif =.. [Op,A1E,A2E],
        E is EKif.


erteke1(K, HL, E) :-
        (   atomic(K) -> helyettesitese(K, HL, E)
        ;   K =.. [Op,A1] ->
            erteke1(A1, HL, A1E),
            EKif =.. [Op,A1E],
            E is EKif
        ;   K =.. [Op,A1,A2] ->
            erteke1(A1, HL, A1E),
            erteke1(A2, HL, A2E),
            EKif =.. [Op,A1E,A2E],
            E is EKif
        ).


erteke2(K, HL, E) :-
        (   atom(K) ->
            (   memberchk(K-V, HL) -> E = V
            ;   E = 0
            )
        ;   number(K) -> E = K
        ;   K =.. [Op,A1] ->
            erteke2(A1, HL, A1E),
            EKif =.. [Op,A1E],
            E is EKif
        ;   K =.. [Op,A1,A2] ->
            erteke2(A1, HL, A1E),
            erteke2(A2, HL, A2E),
            EKif =.. [Op,A1E,A2E],
            E is EKif
        ).

% 7. M2 mintafeladat segédeljárása:

%     Írjon olyan Prolog eljárást, amely egy karakterkódokból álló listát két
%     részre vág! Az első rész legyen a lista olyan maximális hosszú kezdőszelete,
%     amelyben minden elem egy ASCII kisbetű kódja, feltéve, hogy ez a kezdőszelet
%     legalább kételemű. A másik rész legyen a lista fennmaradó része. Ha a lista
%     nem kisbetű-kóddal kezdődik, vagy csak egyetlen kisbetű-kód áll az elején,
%     akkor az eljárás hiúsuljon meg!


% kezdo_szava(+L, ?Kezdet, ?Maradek): Kezdet az L karakterkód-lista maximális
% hosszú csak kisbetű-kódokat tartalmazó kezdőszelete, amely legalább
% kételemű.  Maradek az L-ben Kezdet után álló elemek listája.

kezdo_szava(L, Kezdet, Maradek) :-
        ksz(L, Kezdet, Maradek),
        Kezdet = [_,_|_].

% ksz(+L, ?K, ?M): K az L karakterkód-lista maximális hosszú csak
% kisbetű-kódokat tartalmazó nem-üres kezdőszelete (amely akár 0 vagy 1
% elemű is lehet).  M az L-ben K után álló elemek listája.
ksz(L, K, M) :-
        (   L = [C|L1], kisbetu(C)
        ->  K = [C|K1],
            ksz(L1, K1, M)
        ;   K = [], M = L
```

```
        ).

kezdo_szava2(L, Kezdet, Maradek) :-
        (   append(K, M, L),
%           \+ (  member(C, K), \+ kisbetu(C)  ),
            \+ (  M = [C|_], kisbetu(C)  )
        ->  K = [_,_|_], Kezdet = K, Maradek = M
        ).

% kisbetu(C) : C egy ASCII kisbetű kódja.
kisbetu(C) :-
    C >= 0'a, C =< 0'z.


% 8. M2 teljes feladat:

%     A kezdo_szava/3 predikátum segítségével írjon olyan Prolog eljárást, amely
%     egy adott atomban keres egy abban előforduló legalább kétkarakteres olyan
%     folytonos részatomot, amely csupa kisbetűből áll, és maximális, azaz egyik
%     irányban sem terjeszthető ki kisbetűvel! Az eljárás adja ki a megtalált
%     részatom kezdő indexpozícióját is (1-től számozva)!  Visszalépéskor legyen
%     hajlandó az összes ilyen részatomot felsorolni!  A szavakat az előfordulásuk
%     sorrendjében sorolja fel!

% Egyszerűsített feladat, szava/2:

% szava(Atom, Szo): Az Atom atomban valamely kezdőpozíción a Szo
% áll, amely csupa kisbetűből álló maximális, legalább kétbetűs atom.
szava(Atom, Szo) :-
        atom_codes(Atom, AtomL),
        lista_szava(AtomL, SzoL),
        atom_codes(Szo, SzoL).

% szava(AtomL, SzoL): Az AtomL kódlistában valamely pozíción a SzoL kódlista
% kezdődik, amely csupa kisbetűből áll, maximális és legalább kétbetűs.
lista_szava(AL, SL) :-
        (   kezdo_szava(AL, SL1, ML)
        ->  (   SL = SL1
            ;   lista_szava(ML, SL)
            )
        ;   AL = [_|AL1],
            lista_szava(AL1, SL)
        ).
                % Mi történik, ha a fenti eljárásban a -> helyett vessző van?


% szava(Atom, Szo, Index): Az Atom atomban az Index kezdőpozíción a Szo
% áll, amely csupa kisbetűből álló maximális, legalább kétbetűs atom.
szava(Atom, Szo, Index) :-
        atom_codes(Atom, AtomL),
        lista_szava(AtomL, SzoL, 1, Index),
        atom_codes(Szo, SzoL).

% szava(AtomL, SzoL): Az AtomL kódlistában az Index pozíción a SzoL kódlista
% kezdődik, amely csupa kisbetűből áll, maximális és legalább kétbetűs.
lista_szava(AL, SL, I0, I) :-
        (   kezdo_szava(AL, SL1, ML)
        ->  (   SL = SL1, I = I0
            ;   length(SL1, Len), I1 is I0+Len,
                lista_szava(ML, SL, I1, I)
            )
        ;   AL = [_|AL1], I1 is I0+1,
            lista_szava(AL1, SL, I1, I)
        ).
```