```
% Programs from Tutorial 3 (21/9 2015)


% longer_( Xs, Ys ) − Xs, Ys are lists, Xs has more elements than Ys

longer_( [_|L], [] ) :− list( L ).
longer_( [_|Xs], [_|Ys] ) :− longer_( Xs, Ys ).

% list( L ) − L is a list

list( [] ).
list( [_|T] ) :− list( T ).


% longer( Xs, Ys ) −  Ys is a list, and if Xs is a list then
%                     Xs has more elements than Ys
longer( [_|_], [] ).
longer( [_|Xs], [_|Ys] ) :− longer( Xs, Ys ).

% % %


% append4...( X, Y, Z, XYZ ) − XYZ is the result of appending lists X, Y, Z

append4( X, Y, Z, XYZ ) :− append( X, Y, XY ), append( XY, Z, XYZ).
%
% Loops when called with variables as the first three arguments
% (after producing the answers)

append41( X, Y, Z, XYZ ) :− append( X, YZ, XYZ), append( Y, Z, YZ).
%
% Correctly joins 3 lists, and splits a list.
% More efficient for joining 3 lists:  X passed only once
%                                      instead of twice.

% A more precise specification:
% append4...( X, Y, Z, XYZ ) −
%   X, Y, Z, XYZ are lists, and XYZ is the result of appending lists X, Y, Z,
%   provided that Z or XYZ is a list

% % %


% notmember( _X, Xs ) :− Xs is a list which does not contain X as an element

notmember( _X, [] ).
notmember( X, [Y|Ys] ) :− dif( X, Y ), notmember( X, Ys ).

% Remember that dif/2 checks if its arguments are distinct,
% but is not selected until a correct check is possible.

% % %


% replace1( E, L, Enew, Lnew ) − E is an element of list L and
%         Lnew is L with element E replaced by Enew.
%         (One occurrence of E is replaced.)

replace1( E, [E|T], Enew, [Enew|T] ).
replace1( E, [H|T], Enew, [H|Tnew] ) :− replace1( E, T, Enew, Tnew ).

% A more precise description:
% replace1( E, L, Enew, Lnew ) − If L is a list then E is its element
%         Lnew is L with one occurrence of E replaced by Enew.
```

```
% Replacing more (or 0) occurrences
% replace( E, L, Enew, Lnew ) − L, Lnew are lists, and
%                Lnew is L with some occurrences of E replaced by Enew

replace( _, [],  _, [] ).
replace( E, [E|T], Enew, [Enew|Tnew] ) :−
        replace( E, T, Enew, Tnew ).
replace( E, [H|T], Enew, [H|Tnew] ) :−
        replace( E, T, Enew, Tnew ).

% Replacing all the occurrences
% replaceall( E, L, Enew, Lnew ) − L, Lnew are lists, and
%                Lnew is L with each occurrence of E replaced by Enew

replaceall( _, [],  _, [] ).
replaceall( E, [E|T], Enew, [Enew|Tnew] ) :−
        replaceall( E, T, Enew, Tnew ).
replaceall( E, [H|T], Enew, [H|Tnew] ) :− dif(E,H),
        replaceall( E, T, Enew, Tnew ).

/* Example queries
   replace1( 1, [1,2,3,1,2,1,3], a, L )   − 3 answers
   replace( 1, [1,2,3,1,2,1,3], a, L )    − 8 answers
   replaceall( 1, [1,2,3,1,2,1,3], a, L ) − 1 answer
*/
```