```
% Exista last(?Elem, ?List)
my_last(X,[X]).
my_last(X,[_|L]) :- my_last(X,L).
```

```
swap([X, Y|T], [Y, X|T]):-X>Y.
swap([Z|T], [Z|TT]):-swap(T, TT).
```

```
busort(L,S):-swap(L,LS),!,busort(LS, S).
busort(S,S).
```

```
last_but_one(X,[X,_]).
last_but_one(X,[_,Y|Ys]) :-
last_but_one(X,[Y|Ys]).
```

```
cmmdc(X,0,X) :- X > 0.
cmmdc(X,Y,G) :- Y > 0, Z is X mod
Y, cmmdc(Y,Z,G).
```

```
tree(1,t(a,t(b,t(d,nil,nil),
t(e,nil,nil)),t(c,nil,t(f,t(g,nil,nil),nil)))).
tree(2,t(a,nil,nil)).
tree(3,nil).
```

```
element_at(X,[X|_],1).
element_at(X,[_|L],K) :- K > 1, K1 is K - 1,
element_at(X,L,K1).
```

```
istree(nil).
istree(t(_,L,R)) :- istree(L),
istree(R).
```

```
symmetric(nil).
symmetric(t(_,L,R)) :- mirror(L,R).
```

```
my_length([],0).
my_length([_|L],N) :- my_length(L,N1), N is
N1 + 1.
```

```
min([X], X):-!.
min([P|R], P):-min(R,X), X > P, !.
min([P|R],X):-min(R,X), X =< P.
```

```
mirror(nil,nil).
mirror(t(_,L1,R1),t(_,L2,R2)) :-
mirror(L1,R2), mirror(R1,L2).
```

```
my_reverse(L1,L2) :- my_rev(L1,L2,[]).
my_rev([],L2,L2) :- !.
my_rev([X|Xs],L2,Acc) :-
my_rev(Xs,L2,[X|Acc]).
```

```
split(L,0,[],L).
split([X|Xs],N,[X|Ys],Zs) :- N > 0,
N1 is N - 1, split(Xs,N1,Ys,Zs).
```

```
remove_at(X,[X|Xs],1,Xs).
remove_at(X,[Y|Xs],K,[Y|Ys]) :- K > 1,
  K1 is K - 1, remove_at(X,Xs,K1,Ys).
```

```
is_palindrome(L) :- reverse(L,L).
```

```
insert_at(X,L,K,R) :-
remove_at(X,R,K,L).
```

```
min-sort([], []):-!.
min-sort(L, [M|LS]):-min(L, M),
select(M, L, LM), min-sort(LM, LS).
```

```
my_flatten(X,[X]) :- \+ is_list(X).
my_flatten([],[]).
my_flatten([X|Xs],Zs) :-
my_flatten(X,Y), my_flatten(Xs,Ys),
append(Y,Ys,Zs).
```

```
drop(L1,N,L2) :- drop(L1,N,L2,N).
drop([],_,[],_).
drop([_|Xs],N,Ys,1) :-
drop(Xs,N,Ys,N).
drop([X|Xs],N,[X|Ys],K) :- K > 1, K1
is K - 1, drop(Xs,N,Ys,K1).
```

```
slice([X|_],1,1,[X]).
slice([X|Xs],1,K,[X|Ys]) :- K > 1,
  K1 is K - 1, slice(Xs,1,K1,Ys).
slice([_|Xs],I,K,Ys) :- I > 1,
  I1 is I - 1, K1 is K - 1,
slice(Xs,I1,K1,Ys).
```

```
compress([],[]).
compress([X],[X]).
compress([X,X|Xs],Zs) :-
compress([X|Xs],Zs).
compress([X,Y|Ys],[X|Zs]) :- X \= Y,
compress([Y|Ys],Zs).
```

```
decode([],[]).
decode([X|Ys],[X|Zs]) :- \+
is_list(X), decode(Ys,Zs).
decode([[1,X]|Ys],[X|Zs]) :-
decode(Ys,Zs).
decode([[N,X]|Ys],[X|Zs]) :- N > 1,
N1 is N - 1, decode([[N1,X]|Ys],Zs).
```

```
count(X,[],[],1,X).
count(X,[],[],N,[N,X]) :- N > 1.
count(X,[Y|Ys],[Y|Ys],1,X) :- X \= Y.
count(X,[Y|Ys],[Y|Ys],N,[N,X]) :- N >
1, X \= Y.
count(X,[X|Xs],Ys,K,T) :- K1 is K + 1,
count(X,Xs,Ys,K1,T).
```

```
transfer(X,[],[],[X]).
transfer(X,[Y|Ys],[Y|Ys],[X]) :- X \= Y.
transfer(X,[X|Xs],Ys,[X|Zs]) :-
transfer(X,Xs,Ys,Zs).
```

```
dupli([],[]).
dupli([X|Xs],[X,X|Ys]) :-
dupli(Xs,Ys).
```

```
encode_direct([],[]).
encode_direct([X|Xs],[Z|Zs]) :-
count(X,Xs,Ys,1,Z),
encode_direct(Ys,Zs).
```

```
pack([],[]).
pack([X|Xs],[Z|Zs]) :-
transfer(X,Xs,Ys,Z), pack(Ys,Zs).
```

```
range(I,I,[I]).
range(I,K,[I|L]) :- I < K, I1 is I + 1,
range(I1,K,L).
```

```
has_factor(N,L) :- N mod L =:= 0.
has_factor(N,L) :- L * L < N, L2 is L + 2,
has_factor(N,L2).
```

```
transform([],[]).
transform([[X|Xs]|Ys],[[N,X]|Zs]) :-
length([X|Xs],N), transform(Ys,Zs).
```

```
combination(0,_,[]).
combination(K,L,[X|Xs]) :- K > 0,
  el(X,L,R), K1 is K-1,
combination(K1,R,Xs).
```

```
is_prime(2).
is_prime(3).
is_prime(P) :- integer(P), P > 3, P mod
2 =\= 0, \+ has_factor(P,3).
```

```
encode(L1,L2) :- pack(L1,L),
transform(L,L2).
```

```
el(X,[X|L],L).
el(X,[_|L],R) :- el(X,L,R).
```

```
qsort([], []).
qsort([H|T], S):-split(H, T, L,
R),qsort(L, LS), qsort(R, RS),
append(LS, [H|RS], S).
```

```
strip([],[]).
strip([[1,X]|Ys],[X|Zs]) :- strip(Ys,Zs).
strip([[N,X]|Ys],[[N,X]|Zs]) :- N > 1,
strip(Ys,Zs).
```

```
rotate([],_,[]) :- !.
rotate(L1,N,L2) :-
  length(L1,NL1), N1 is N mod NL1,
split(L1,N1,S1,S2),
append(S2,S1,L2).
```

```
rnd_select(_,0,[]).
rnd_select(Xs,N,[X|Zs]) :- N > 0,
  length(Xs,L), I is random(L) + 1,
  remove_at(X,Xs,I,Ys),
  N1 is N - 1, rnd_select(Ys,N1,Zs).
```

```
encode_modified(L1,L2) :-
encode(L1,L), strip(L,L2).
```

```
lotto(N,M,L) :- range(1,M,R),
rnd_select(R,N,L).
```

```
rnd_permu(L1,L2) :- length(L1,N),
rnd_select(L1,N,L2).
```