```
replace([], _, _, []) :- !.
replace([Item1|L1], Item1, Item2, [Item2|R2]) :- replace(L1, Item1, Item2, R2), !.
replace([X|L1], Item1, Item2, [X|R2]) :- replace(L1, Item1, Item2, R2).

reunion([], L2, L2).
reunion(L1, [], L1).
reunion([X|L1], [Y|L2], [X|L3]) :- X < Y, !, reunion(L1, [Y|L2], L3).
reunion([X|L1], [Y|L2], [Y|L3]) :- X > Y, !, reunion([X|L1], L2, L3).
reunion([X|L1], [X|L2], [X|L3]) :- reunion(L1, L2, L3), !.

par(X) :- mod(X, 2) =:= 0.
filter(L1, L2) :- findall(X, (member(X, L1), par(X)), L2).
pare(L1, L2) :- filter(L1, L2).

functie2(X, Acc, [X|Acc]).
myfold([], Acc, Acc).
myfold([X|L1], Acc, Y) :- \+ member(X, Acc), !, functie2(X, Acc, Z), myfold(L1, Z, Y).
myfold([X|L1], Acc, Y) :- member(X, Acc), myfold(L1, Acc, Y).
set([], []).
set(L1, L2) :- myfold(L1, [], L2).

functie1(X, Acc, Y) :- Y is X + Acc.
myfold([], Acc, Acc).
myfold([X|L1], Acc, Y) :- functie1(X, Acc, Z), myfold(L1, Z, Y).

zipWith([], [], []).
zipWith([X|L1], [Y|L2], [Z|L3]) :- functie1(X, Y, Z), zipWith(L1, L2, L3).

functie(X, Y) :- Y is 2 * X.
mymap([], []).
mymap([X|L1], [Y|L2]) :- functie(X, Y), mymap(L1, L2).

orMap([]) :- fail.
orMap([X|L1]) :- par(X) ; orMap(L1), !.

andMap([]).
andMap([X|L1]) :- par(X), andMap(L1), !.

reverse(X, L) :- reverse(X, L, []).
reverse([],Z,Z).
reverse([H|T],Z,Acc) :- reverse(T,Z,[H|Acc]).

cons(X, L, [X|L]).
myFoldR(L, Acc, R) :- reverse(L, L1), myFoldL(L1, Acc, R).
myFoldL([], Acc, Acc).
myFoldL([X|L1], Acc, Y) :- cons(X, Acc, Z), myFoldL(L1, Z, Y).

adjacent(X, Y, Zs) :- append(_, [X, Y|_], Zs), !.
```

```
last(X, Xs) :- append(_, [X], Xs), !.
subsequence([X|Xs], [X|Ys]) :- subsequence(Xs, Ys).
subsequence(Xs, [_|Ys]) :- subsequence(Xs, Ys).
subsequence([], _) :- !.
double([], []).
double([X|L1], [X, X|L2]) :- double(L1, L2).
palindrome([]).
palindrome([_]).
palindrome(Pal) :- append([H|T], [H], Pal), palindrome(T), !.

take(1, [H|_], H) :- !.
take(N, [_|T], X) :- N1 is N-1, take(N1, T, X).

zipWith([], [], []).
zipWith([X|L1], [Y|L2], [Z|L3]) :- functie1(X, Y, Z), zipWith(L1, L2, L3).

zip([], [], []).
zip([X|L1], [Y|L2], [(X, Y)|L3]) :- zip(L1, L2, L3).
```