↦ да добавим X към L и да получим N
=

add(X, L, N).

member(X, L).

% e - празен списък

f(H, T). -▷ глава и опашка → [H|T] → шаблон

[1] ▷ f(1, e), [ ] - празен

add(X, L, f(X, L)). → [1|[ ]]

member(X, f(X, Y)).

member(X, f(X1, Y)) :- member(X, Y).

[1, 2] ◁▷ [1|[2|[ ]]]

синтактична захар

---

add(X, L, [X|L]).

member(X, [X|_]).

member(X, [_|L]) :- member(X, L).

% first(F, L).

first(X, [X|_]).

second(S, [_|[S|_]]). ◁▷ second(S, [_|T]) :- first(S, T).

◁▷ second(S, [_, S|_]).

% last(S, L). → ??

% append(A, B, AB).

12.10.2016 | Предикатът не връща стойността, удовлетворява се, ако е Т.

%last(S, L). - ВЗИМАНЕ НА ПОСЛЕДЕН ЕЛЕМЕНТ.

[H] - последният е H

[H|T] - последният е последния на T

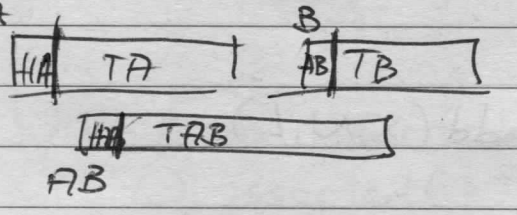
```
last(H,[H]).
last(S,[_|T]):-last(S,T).
```

] *

За предпоследен разглеждането е същото, но приключва при списък с 2 елемента.

* Ако ги разменим, ще работи малко по-бързо в общия случай

```
%append(A,B,AB).
```



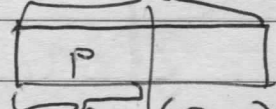
HAB = HA
TAB → TA ∪ B

```
append([],B,B).
append([H|A],B,[H|C]):-append(A,B,C).
```

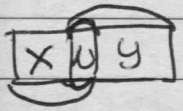
```
?-append([1],[2,3],[1,2,3]).
?-append([1],[2,3],X).
?-append(X,[2,3],[1,2,3]).
   ... [1], X ...
?-append(X,Y,[1,2,3]).
        X = []
        Y = [1,2,3],
    Може и още.
```

Дефиниране на някои предикати чрез append

```
last(S,L):-append(X,[S],L).
member(X,L):-append(_,[X|_],L).
```




```
prefix(P,L):-append(P,_,L).
suffix(S,L):-append(_,S,L).
```

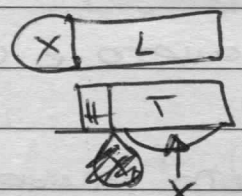



```
infix(N,L):-append(X,N,P),append(P,Y,L).
```

безбр. ← крайн бр.

=> infix(N,L):-append(P,Y,L), append(X,N,P).

%add(X, L, N).



add(X,L,[X|L]).
add(X,[H|T],[H|N]):-add(X,T,N)

%remove(X,L,N)
remove(X,L,N):-add(X,N,L).

---

add(X,L,N):-append(F,B,L), append(F,[X|B],N). -> Тогава remove чрез add има опасност да зацикли.

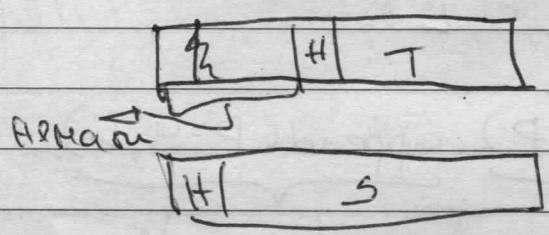
~~=> remove(X,N,L):-append~~

?-remove(1,[1,2,3],X).
?-remove(X,[1,2,3],Y).

Пермутации на списък

perm([],[]).
perm([H|P],L):-~~member(H,L)~~, remove(H,L,L1), perm(P,L1).

%subset(S,L) ! Генериране, не проверяване!
%subset(S,L):-perm(P,L), prefix(S,P).
subset([],L).
subset(S,[H|T]):-subset(S,T).
subset([H|S],[H|T]):-subset(S,T).



3) subset([], L).
subset([H|S], L) :- append(_, [H|T], L), subset(S, T).

Сортировка
%issorted(S).
issorted([]).
issorted([_]).
issorted([A,B|T]) :- A =< B, issorted([B|T])
psort(L,S) :- perm(S,L), issorted(S). // не е ефективно

---

not(A).
$\forall x \varphi \Leftrightarrow \neg\exists x \neg\varphi$
issorted(L) :- not(notsorted(L)).
notsorted(L) :- infix([A,B], L), A > B.
not(A =< B).

X - списък от числа, Y - списък от списъци от числа.
1. p1(X,Y) $\Leftrightarrow$ $\exists$ ел-т на X, който е в ел-т на Y
2. p2(X,Y) $\Leftrightarrow$ $\exists$ ел-т на X, който е във $\forall$ ел-т на Y
3. p3(X,Y) $\Leftrightarrow$ $\forall$ ел-т на X е в ел-т на Y $\forall x \exists y \varphi$
4. p4(X,Y) $\Leftrightarrow$ $\forall$ ел-т на X е в $\forall$ ел-т на Y. $\neg\exists x \forall y \neg\varphi$
$\neg\exists x \neg\exists y \varphi$

p1(X,Y) :- member(A,X), member(B,Y), member(A,B).
p2(X,Y) :- member(A,X), not(member(B,Y), not(member(A,B))).
p3(X,Y) :- not(member(A,X), member(B,Y), member(A,B))).
not($\forall x \forall y \varphi$
$\neg\exists x \exists y \neg\varphi$
p4(X,Y) :-
not(member(A,X), member(B,Y), not(member(A,B))).

Опитай да решиш някое от 2-3 без not.