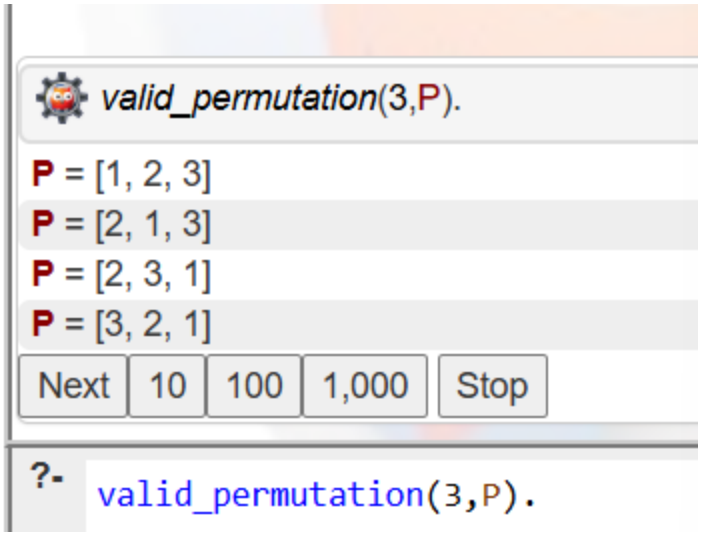
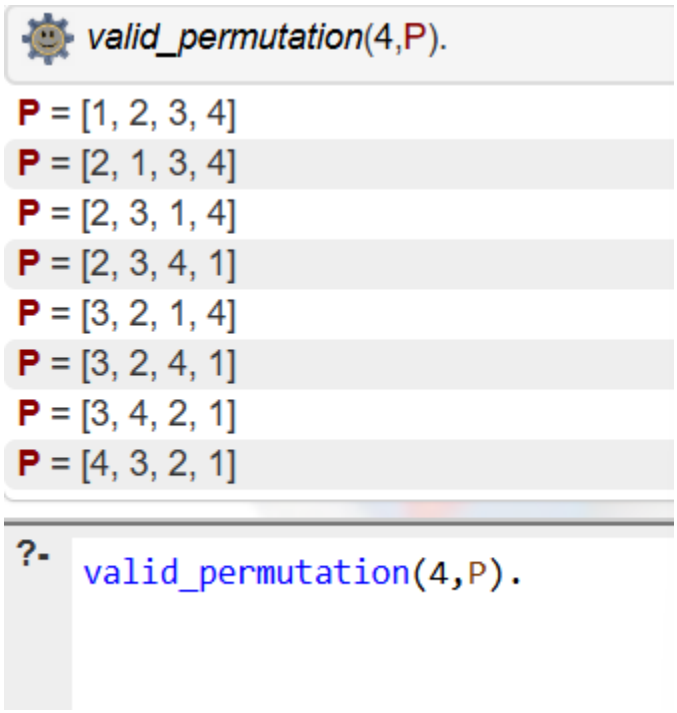
P3 - FILIMON BRIANA, 923

9. Generate all permutation of N (N - given) respecting the property: for every 2<=i<=n exists an 1<=j<=i, so |v(i)-v(j)|=1

```
% Student exercise profile
:- set_prolog_flag(occurs_check, error).        % disallow cyclic terms
:- set_prolog_stack(global, limit(8 000 000)).  % limit term space (8Mb)
:- set_prolog_stack(local,  limit(2 000 000)).  % limit environment space

% Your program goes here
%
valid_permutation(N, Perm) :-
    numlist(1, N, Numbers),
    permute(Numbers, [], RevPerm),
    reverse(RevPerm, Perm).

permute([], Perm, Perm).
permute(Available, Current, Perm) :-
    select(X, Available, Rest),
    ( Current = [] -> true
    ; has_neighbor(X, Current)
    ),
    permute(Rest, [X|Current], Perm).

has_neighbor(X, List) :-
    member(Y, List),
    Diff is abs(X - Y),
    Diff =:= 1.



/** <examples> Your example queries go here, e.g.
?- member(X, [cat, mouse]).
*/
```





Mathematical Model

% valid_permutation(N, Perm) - generates valid permutations of [1..N]
% valid_permutation(N, Perm) =
% Let Numbers = [1, 2, ..., N]
% permute(Numbers, [], RevPerm) ∧ reverse(RevPerm, Perm)

% permute(Available, Current, Perm) - builds permutation with neighbor constraint
% permute([], Current, Perm) = Perm = Current (base case)
% permute([a1, ..., ak], Current, Perm) =
% ∃ ai ∈ [a1, ..., ak] such that:
% (Current = [] ∨ has_neighbor(ai, Current)) ∧

```
%     permute([a1, ..., ai-1, ai+1, ..., ak], [ai | Current], Perm)

% has_neighbor(X, List) - checks if X has an adjacent value in List
% has_neighbor(X, [l1, ..., lm]) =
%   ∃ lj ∈ [l1, ..., lm] : |X - lj| = 1
```