

```
% simple solution to nqueens as example of generate and test method
nqueens(N,Q):-integers(1,N,[],Ints),perm(Ints,Q),safe(Q).

safe([]).
safe([H|T]):-safe(T),not(attack(H,T)).

attack(H,T):-diag(H,1,T).

diag(Q,N,[H|T]):-Q,=: H + N.
diag(Q,N,[H|T]):-Q,=: H - N.
diag(Q,N,[H|T]):-N2 is N + 1, diag(Q,N2,T).

integers(N,N,Sofar,[N|Sofar]):-!.
integers(N,M,Sofar,List):-N2 is N + 1, integers(N2,M,[N | Sofar],List).

perm([],[]).
perm(L,[H|T]):-remove(H,L,L2),perm(L2,T).

remove(H,[H|T],T).
remove(X,[H|T],[H|T2]):-remove(X,T2,T).
```

Integers

```
6 ?- integers(1,3,[],!).
Call: (8) integers(1, 3, [], _G664) ? creep
^ Call: (9) _L176 is 1+1 ? creep
^ Exit: (9) 2 is 1+1 ? creep
Call: (9) integers(2, 3, [], _G664) ? creep
^ Call: (10) _L196 is 2+1 ? creep
^ Exit: (10) 3 is 2+1 ? creep
Call: (10) integers(3, 3, [2, 1], _G664) ? creep
Exit: (10) integers(3, 3, [2, 1], [3, 2, 1]) ? creep
Exit: (9) integers(2, 3, [1], [3, 2, 1]) ? creep
Exit: (8) integers(1, 3, [], [3, 2, 1]) ? creep
I = [3, 2, 1].
```

Perm

```
• 4 ?- perm([a,b,c],N).
• Call: (7) perm([a, b, c], _G830) ? creep
• Call: (8) remove_G903, [a, b, c], _L176 ? creep
• Exit: (8) remove(a, [a, b, c], [b, c]) ? creep
• Call: (8) perm([b, c], _G904) ? creep
• Call: (9) remove_G906, [b, c], _L215 ? creep
• Exit: (9) remove(b, [b, c], [c]) ? creep
• Call: (9) perm([c], _G907) ? creep
• Call: (10) remove_G909, [c], _L254 ? creep
• Exit: (10) remove(c, [c], []) ? creep
• Call: (10) perm([], _G910) ? creep
• Exit: (10) perm([], []) ? creep
• Exit: (9) perm([c], [c]) ? creep
• Exit: (8) perm([b, c], [b, c]) ? creep
• Exit: (7) perm([a, b, c], [a, b, c]) ? creep
• N = [a, b, c].

• Redo: (10) perm([], _G910) ? creep
• Call: (11) remove_G912, [], _L293 ? creep
• Fail: (11) remove_G912, [], _L293 ? creep
• Redo: (10) remove_G909, [c], _L254 ? creep
• Call: (11) remove_G909, [], _G913 ? creep
• Fail: (11) remove_G909, [], _G913 ? creep
• Fail: (9) perm([c], _G907) ? creep
• Redo: (9) remove_G906, [b, c], _L215 ? creep
• Call: (10) remove_G906, [c], _G910 ? creep
• Exit: (10) remove(c, [c], []) ? creep
• Exit: (9) remove(c, [b, c], [b]) ? creep
• Call: (9) perm([b], _G907) ? creep
• Call: (10) remove_G912, [b], _L268 ? creep
• Exit: (10) remove(b, [b], []) ? creep
• Call: (10) perm([], _G913) ? creep
• Exit: (10) perm([], []) ? creep
• Exit: (9) perm([b], [b]) ? creep
• Exit: (8) perm([b, c], [c, b]) ? creep
• Exit: (7) perm([a, b, c], [a, c, b]) ? creep
• N = [a, c, b].
```

% better solution to nqueens problem
 % the test is pushed inside the generator, so that unsuccessful
 % solutions are not fully generated.

```
queens(N,Q):-integers(1,N,[],Ints),q*(Ints,[],Q).

q*([],Queens,Queens):-remove(Next,Unplaced,Rest),
not(diag(Next,1,Safe)),
q*(Rest,[Next|Safe],Queens).
```

Fibonacci

1,1,2,3,5,8,13, ... $f(n) = f(n-1) + f(n-2)$

```
% Simple fibonacci
fib(1,1).
fib(2,1).
fib(N,F):-M is N - 1, P is N - 2, fib(M,G),fib(P,H),F is G + H.

% More efficient fibonacci
fib2(1,1).
fib2(2,1).
fib2(N,F):-fib*(N,F,1,1).

fib*(3,F,M,P):-F is M + P.
fib*(N,F,M,P):-N2 is N - 1, Q is M + P, fib*(N2,F,Q,M).
```

```
• fib(4,N).
• Call: (7) fib(4, _G791) ? creep
• Call: (8) _L174 is 4-1 ? creep
• Exit: (8) 3 is 4-1 ? creep
• Call: (8) _L175 is 4-2 ? creep
• Exit: (8) 2 is 4-2 ? creep
• Call: (8) fib(3, _L176) ? creep
• Call: (9) _L195 is 3-1 ? creep
• Exit: (9) 2 is 3-1 ? creep
• Call: (9) _L196 is 3-2 ? creep
• Exit: (9) 1 is 3-2 ? creep
• Call: (9) fib(2, _L197) ? creep
• Exit: (9) fib(2, 1) ? creep
• Call: (9) fib(2, _L198) ? creep
• Exit: (9) fib(1, 1) ? creep
• Call: (9) _L176 is 1+1 ? creep
• Exit: (9) 2 is 1+1 ? creep
• Exit: (8) fib(3, 2) ? creep
• Call: (8) fib(2, _L177) ? creep
• Exit: (8) fib(2, 1) ? creep
• Call: (8) _G791 is 2+1 ? creep
• Exit: (8) 3 is 2+1 ? creep
• Exit: (7) fib(4, 3) ? creep
• N = 3.

• 14 ?- fib2(4,N).
• Call: (8) fib2(4, _G620) ? creep
• Call: (9) fib*(4, _G620, 1, 1) ? creep
• Call: (10) _L195 is 4-1 ? creep
• Exit: (10) 3 is 4-1 ? creep
• Call: (10) _L196 is 1+1 ? creep
• Exit: (10) 2 is 1+1 ? creep
• Call: (10) fib*(3, _G620, 2, 1) ? creep
• Call: (11) _G620 is 2+1 ? creep
• Exit: (11) 3 is 2+1 ? creep
• Exit: (10) fib*(3, 3, 2, 1) ? creep
• Exit: (9) fib*(4, 3, 1, 1) ? creep
• Exit: (8) fib2(4, 3) ? creep
• N = 3.
```