## Code:

```
/* -------- 8-Queens in PROLOG (backtracking + safety) -------- */
% entry: one solution
solve_8_queens(Qs) :-
Qs = [_,_,_,_,_,_,_,_],
permutation([1,2,3,4,5,6,7,8], Qs),
safe_diagonals(Qs).
% entry: list all solutions with backtracking
all_8_queens(Qs) :-
permutation([1,2,3,4,5,6,7,8], Qs),
safe_diagonals(Qs).
% diagonal safety: no two queens attack diagonally
safe_diagonals(Qs) :-
safe_diagonals(Qs, 1).
safe_diagonals([], _).
safe_diagonals([Q|Rest], Row) :-
no_conflict(Q, Rest, Row, 1),
NextRow is Row + 1,
safe_diagonals(Rest, NextRow).
% check conflicts of current queen against the remaining
no_conflict(_, [], _, _).
no_conflict(Q, [Q1|Rest], Row, Offset) :-
% different columns ensured by permutation
abs(Q - Q1) =\= Offset,
NextOffset is Offset + 1,
no_conflict(Q, Rest, Row, NextOffset).
% utility: absolute value (for portability)
abs(X, A) :- X >= 0 -> A is X ; A is -X.
/* ------ Pretty Printer (optional) ------ */
print_board(Qs) :-
nl, write(' +-----------------+'), nl,
print_rows(Qs, 1),
```

```
write(' +------------------+'), nl.
print_rows([], _).
print_rows([Q|Rest], R) :-
writef('%2r |', [R]),
print_cols(1, Q), nl,
R1 is R + 1,
print_rows(Rest, R1).
print_cols(C, Q) :- C =< 8, !,
( C =:= Q -> write(' Q') ; write(' .') ),
C1 is C + 1,
print_cols(C1, Q).
print_cols(_, _) :- write(' |').
```

## Output:

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.
For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
?- ['G:/8queue'].
true.
?- solve_8_queens(Qs).
Qs = [1, 5, 8, 6, 3, 7, 2, 4] ;
Qs = [1, 6, 8, 3, 7, 4, 2, 5] ;
Qs = [1, 7, 4, 6, 8, 2, 5, 3] ;
Qs = [1, 7, 5, 8, 2, 4, 6, 3] ;
Qs = [2, 4, 6, 8, 3, 1, 7, 5] ;
Qs = [2, 5, 7, 1, 3, 8, 6, 4] ;
Qs = [2, 5, 7, 4, 1, 8, 6, 3] ;
Qs = [2, 6, 1, 7, 4, 8, 3, 5] ;
Qs = [2, 6, 8, 3, 1, 4, 7, 5] ;
Qs = [2, 7, 3, 6, 8, 5, 1, 4] ;
```

```
Qs = [2, 7, 5, 8, 1, 4, 6, 3] ;
Qs = [2, 8, 6, 1, 3, 5, 7, 4] ;
Qs = [3, 1, 7, 5, 8, 2, 4, 6] ;
Qs = [3, 5, 2, 8, 1, 7, 4, 6] ;
Qs = [3, 5, 2, 8, 6, 4, 7, 1] ;
Qs = [3, 5, 7, 1, 4, 2, 8, 6] ;
Qs = [3, 5, 8, 4, 1, 7, 2, 6] .
?- solve_8_queens(Qs), print_board(Qs).
+-----------------+
1 | Q . . . . . . . |
2 | . . . . Q . . . |
3 | . . . . . . . Q |
4 | . . . . . Q . . |
5 | . . Q . . . . . |
6 | . . . . . . Q . |
7 | . Q . . . . . . |
8 | . . . Q . . . . |
+-----------------+
Qs = [1, 5, 8, 6, 3, 7, 2, 4] ;
+-----------------+
1 | Q . . . . . . . |
2 | . . . . . Q . . |
3 | . . . . . . . Q |
4 | . . Q . . . . . |
5 | . . . . . . Q . |
6 | . . . Q . . . . |
7 | . Q . . . . . . |
8 | . . . . Q . . . |
+-----------------+
Qs = [1, 6, 8, 3, 7, 4, 2, 5] .
?- all_8_queens(Qs).
Qs = [1, 5, 8, 6, 3, 7, 2, 4] ;
```

Qs = [1, 6, 8, 3, 7, 4, 2, 5] ;

Qs = [1, 7, 4, 6, 8, 2, 5, 3] ;

Qs = [1, 7, 5, 8, 2, 4, 6, 3] ;

Qs = [2, 4, 6, 8, 3, 1, 7, 5] ;

Qs = [2, 5, 7, 1, 3, 8, 6, 4] ;

Qs = [2, 5, 7, 4, 1, 8, 6, 3] ;

Qs = [2, 6, 1, 7, 4, 8, 3, 5] ;

Qs = [2, 6, 8, 3, 1, 4, 7, 5] ;

Qs = [2, 7, 3, 6, 8, 5, 1, 4] .

?- findall(Q, all_8_queens(Q), L), length(L, N).

L = [[1, 5, 8, 6, 3, 7, 2, 4], [1, 6, 8, 3, 7, 4, 2|...], [1, 7, 4, 6, 8, 2|...], [1, 7, 5, 8, 2|...], [2, 4, 6, 8|...], [2, 5, 7|...], [2, 5|...], [2|...], [...|...]|...],

N = 92.