EXPERIMENT 9

AIM : WAP to solve 4-queen problem in PROLOG.

THEORY : The 4-Queens problem is a specific case of the classic N-Queens problem in which the goal is to place 4 queens on a 4×4 chessboard such that no two queens threaten each other. A queen in chess can move any number of squares vertically, horizontally, or diagonally, so the solution must ensure that no two queens share the same row, column, or diagonal. In Prolog, this problem is solved using a **permutation-based backtracking approach**, where each permutation of the list [1, 2, 3, 4] represents a possible placement of queens in different columns, one in each row. The program checks each permutation to determine if it is **safe**, i.e., no two queens are on the same diagonal, by comparing the difference in row and column indices. This approach leverages Prolog's powerful pattern matching and recursion to generate and test all valid configurations efficiently, ultimately producing only those placements where no queens attack each other.

CODE :

```
% Main predicate to solve N-Queens
queens(N, Solution) :-
    range(1, N, Range),
    permutation(Range, Solution),
    safe(Solution).

% Generate list from From to To
range(From, To, [From|Rest]) :-
    From =< To,
    Next is From + 1,
    range(Next, To, Rest).
range(From, To, []) :-
    From > To.

% Check if the board is safe (no queens attacking each other diagonally)
safe([]).
safe([Q|Others]) :-
    safe(Q, Others, 1),
    safe(Others).
```

```
% Check that no two queens attack each other diagonally
safe(_, [], _).
safe(Q, [Q1|Others], D) :-
    Q =\= Q1 + D,
    Q =\= Q1 - D,
    D1 is D + 1,
    safe(Q, Others, D1).
```

QUERY :

```
?- queens(4, Solution).
```

OUTPUT :

```
Solution = [2, 4, 1, 3] ;
Solution = [3, 1, 4, 2] ;
false.
```