Experiment No: 6

Title: Prolog Programming (Knowledge Engineering - Advance) - N Queen Problem

Implementation:

Code:

```
% The predicate n_queens/2 takes the size of the board (N) and returns a list of positions of the queens on the board
n_queens(N, Positions) :-
  range(1, N, Rows), % Create a list of row numbers from 1 to N
   permutation(Rows, Positions), % Generate a permutation of the row numbers to get a list of possible queen positions
   safe_queens(Positions). % Check if the queen positions are safe (i.e., no two queens threaten each other)

% The predicate range/3 creates a list of integers from Min to Max
range(Min, Max, []) :- Min > Max.
range(Min, Max, [Min|Rest]) :-
  Min =< Max,
  Next is Min + 1,
  range(Next, Max, Rest).

% The predicate safe_queens/1 checks if a list of queen positions is safe
safe_queens([]).
safe_queens([Queen|Rest]) :-
  safe_queens(Rest),
  no_attack(Queen, Rest, 1).

% The predicate no_attack/3 checks if a queen at position (Row, Col) can attack any of the queens in Rest
no_attack(_, [], _).
no_attack(Queen, [Y|Rest], Dist) :-
  Queen =\= Y,
  Queen + Dist =\= Y,
  Queen - Dist =\= Y,
  NextDist is Dist + 1,
  no_attack(Queen, Rest, NextDist).

% Use the clpfd library for arithmetic constraints
:- use_module(library(clpfd)).
```

Output:

```
n_queens(8, Positions).

Positions = [1, 5, 8, 6, 3, 7, 2, 4]
Positions = [1, 6, 8, 3, 7, 4, 2, 5]
Positions = [1, 7, 4, 6, 8, 2, 5, 3]
Positions = [1, 7, 5, 8, 2, 4, 6, 3]
Positions = [2, 4, 6, 8, 3, 1, 7, 5]
Positions = [2, 5, 7, 1, 3, 8, 6, 4]
Positions = [2, 5, 7, 4, 1, 8, 6, 3]
Positions = [2, 6, 1, 7, 4, 8, 3, 5]
Positions = [2, 6, 8, 3, 1, 4, 7, 5]
Next | 10 | 100 | 1,000 | Stop

n_queens(4, Positions).

Positions = [2, 4, 1, 3]
Positions = [3, 1, 4, 2]
false
```