Experiment-5:

Implementing the prolog program for N-Queen Problem.

Objective:

We have to create a prolog cod for N-Queen Problem and then checking the validation of this code by providing different values of N.

N-Queen Problem:

```
use_module(library(lists)).


n_queen(N, Solution) :-

        length(Solution, N),

        queen(Solution, N).



up2N(N,N,[N]) :-!.
up2N(K,N,[K|Tail]) :- K < N, K1 is K+1, up2N(K1, N, Tail).


queen([],_).

queen([Q|Qlist],N) :-
```

```
    queen(Qlist, N),

    up2N(1,N,Candidate_positions_for_queenQ),

    member(Q, Candidate_positions_for_queenQ),

    check_solution(Q,Qlist, 1).

check_solution(_,[], _).
```

Output:

```
1 ?-
% u:/4-1/artificial intelligence lab/lab5/nqueen compiled 0.00 sec, 9 clauses
1 ?- n_queen(4,S).
S = [3, 1, 4, 2] .

2 ?- n_queen(8,S).
S = [4, 2, 7, 3, 6, 8, 5, 1] .

3 ?- n_queen(2,S).
false.

4 ?- n_queen(4,S).
S = [3, 1, 4, 2] .

5 ?-
```