Fr. Conceicao Rodrigues College of Engineering Fr.
Agnel Ashram, Bandstand, Bandra (W), Mumbai -
400050

Department of Computer Engineering

Academic Term II: 23-24

Class: B.E (Computer), Sem – VI Subject Name: Artificial Intelligence Student

Name: Pushpendersingh Bisht Roll No: 9526

Practical No:	**8**
Title:	Programming in PROLOG
Date of Performance:	**25/03/2024**
Date of Submission:	**01/04/2024**

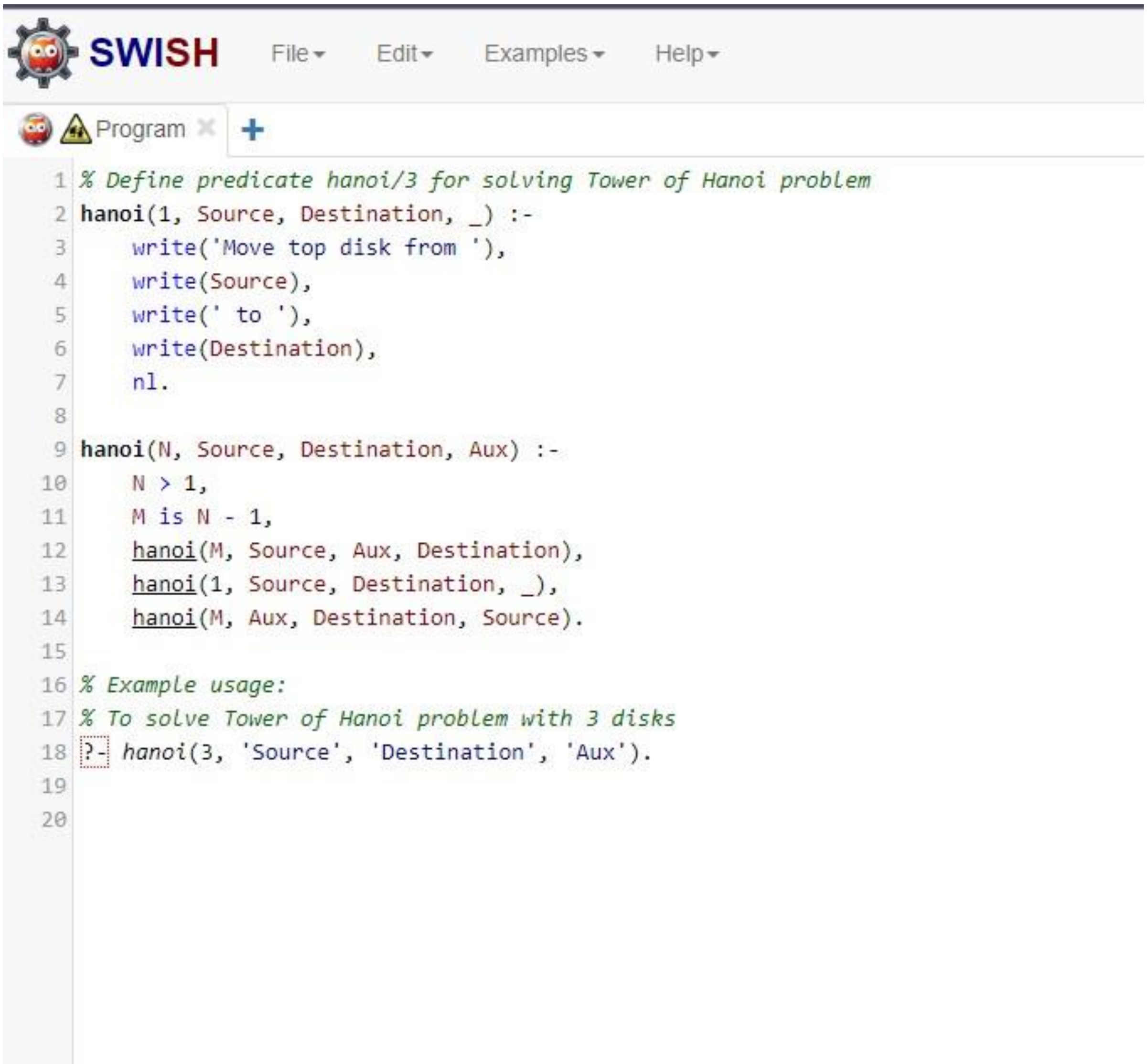
Rubrics for Evaluation:

Sr. No	**Performance Indicator**	**Excellent**	**Good**	**Below Average**	**Marks**
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Logic/Algorithm Complexity analysis (03)	03(Correct)	02(Partial)	01 (Tried)	
3	Coding Standards (03): Comments/indention/Naming conventions Test Cases /Output	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Assignment (03)	03(done well)	2 (Partially Correct)	1(submitted)	
Total					

Signature of the Teacher:

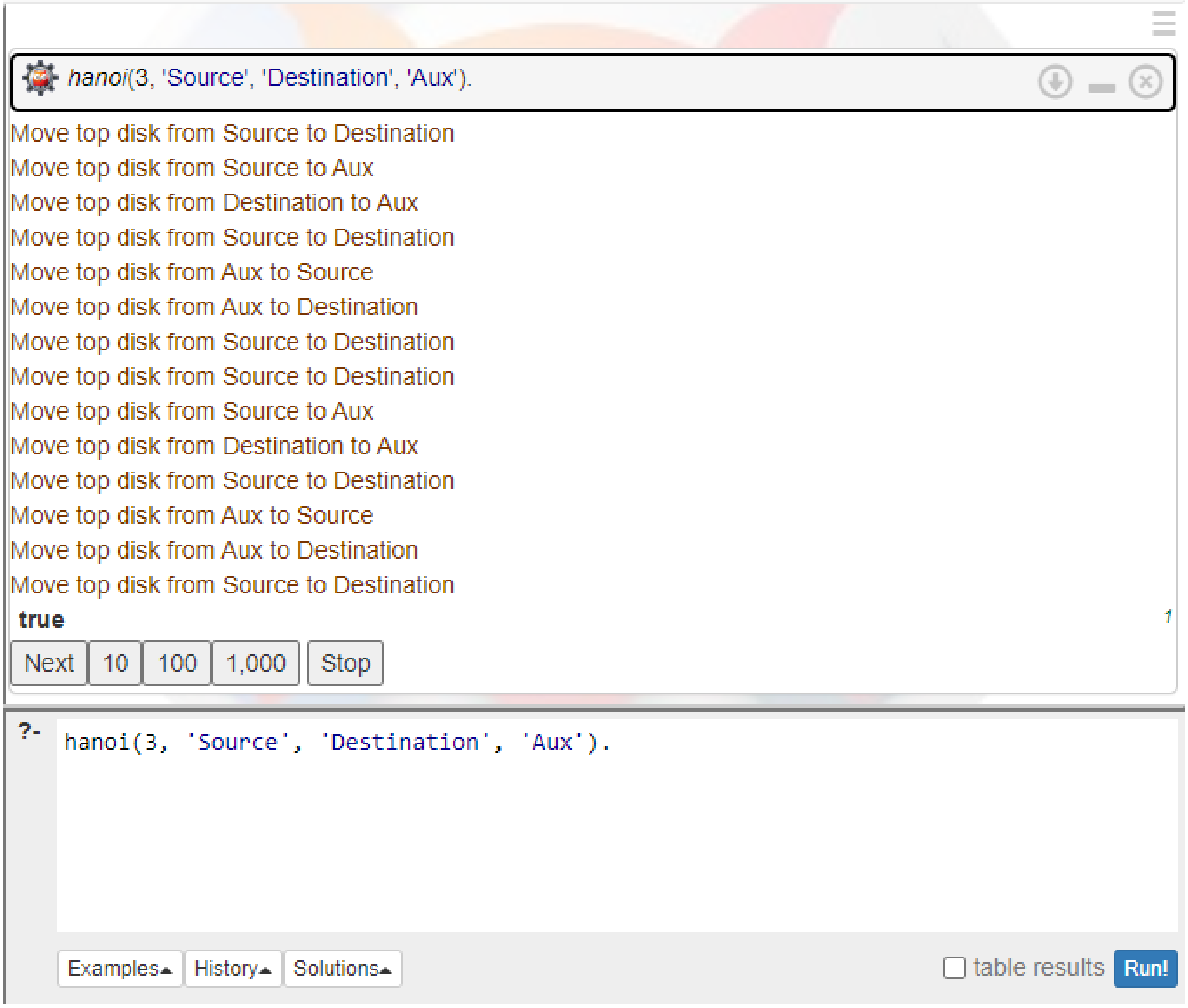
A) Tower of Hanoi

Source code:



```prolog
% Define predicate hanoi/3 for solving Tower of Hanoi problem
hanoi(1, Source, Destination, _) :-
    write('Move top disk from '),
    write(Source),
    write(' to '),
    write(Destination),
    nl.

hanoi(N, Source, Destination, Aux) :-
    N > 1,
    M is N - 1,
    hanoi(M, Source, Aux, Destination),
    hanoi(1, Source, Destination, _),
    hanoi(M, Aux, Destination, Source).

% Example usage:
% To solve Tower of Hanoi problem with 3 disks
?- hanoi(3, 'Source', 'Destination', 'Aux').
```

Output:

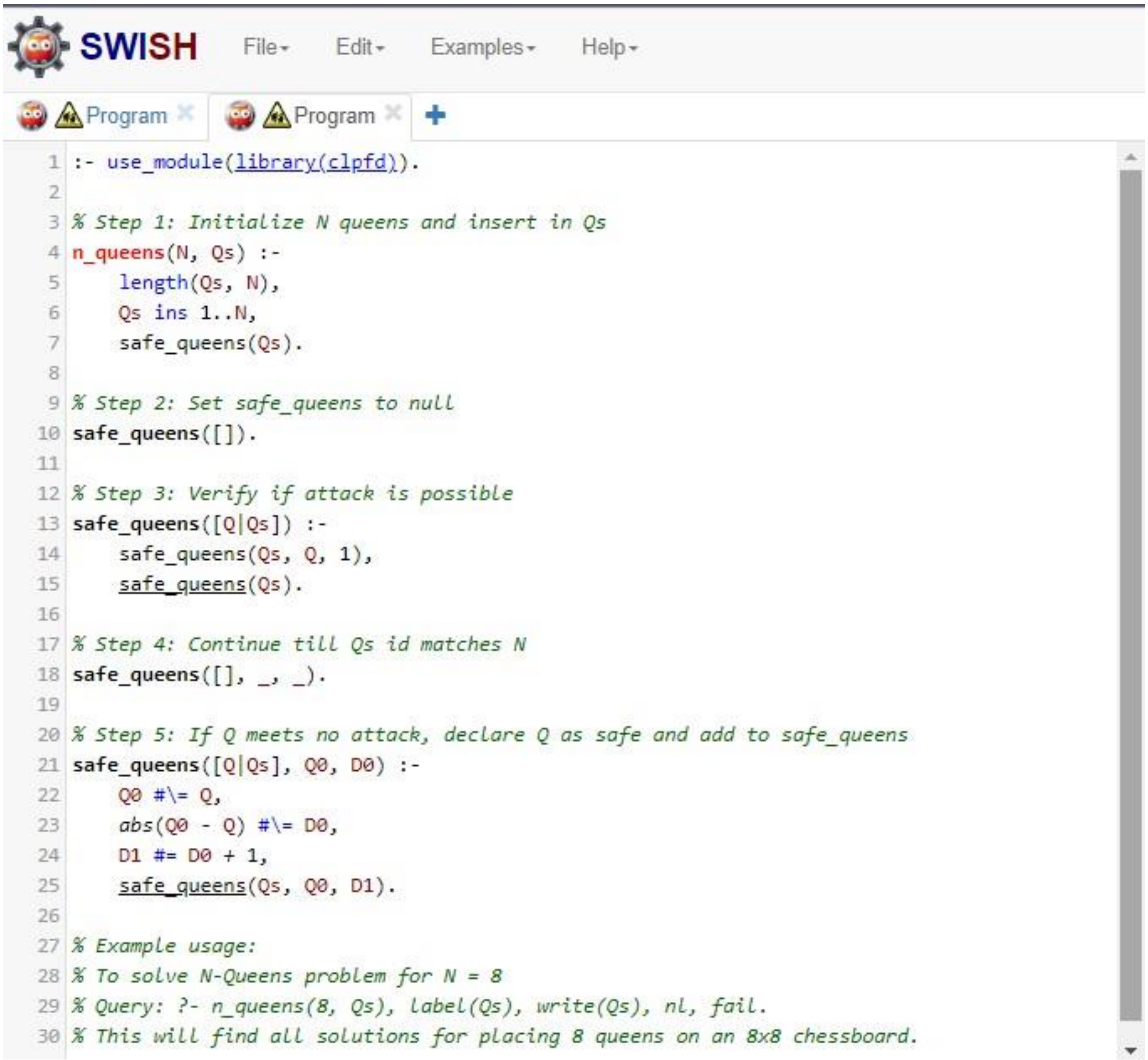


```
hanoi(3, 'Source', 'Destination', 'Aux').
Move top disk from Source to Destination
Move top disk from Source to Aux
Move top disk from Destination to Aux
Move top disk from Source to Destination
Move top disk from Aux to Source
Move top disk from Aux to Destination
Move top disk from Source to Destination
Move top disk from Source to Destination
Move top disk from Source to Aux
Move top disk from Destination to Aux
Move top disk from Source to Destination
Move top disk from Aux to Source
Move top disk from Aux to Destination
Move top disk from Source to Destination
true
```

```
?- hanoi(3, 'Source', 'Destination', 'Aux').
```

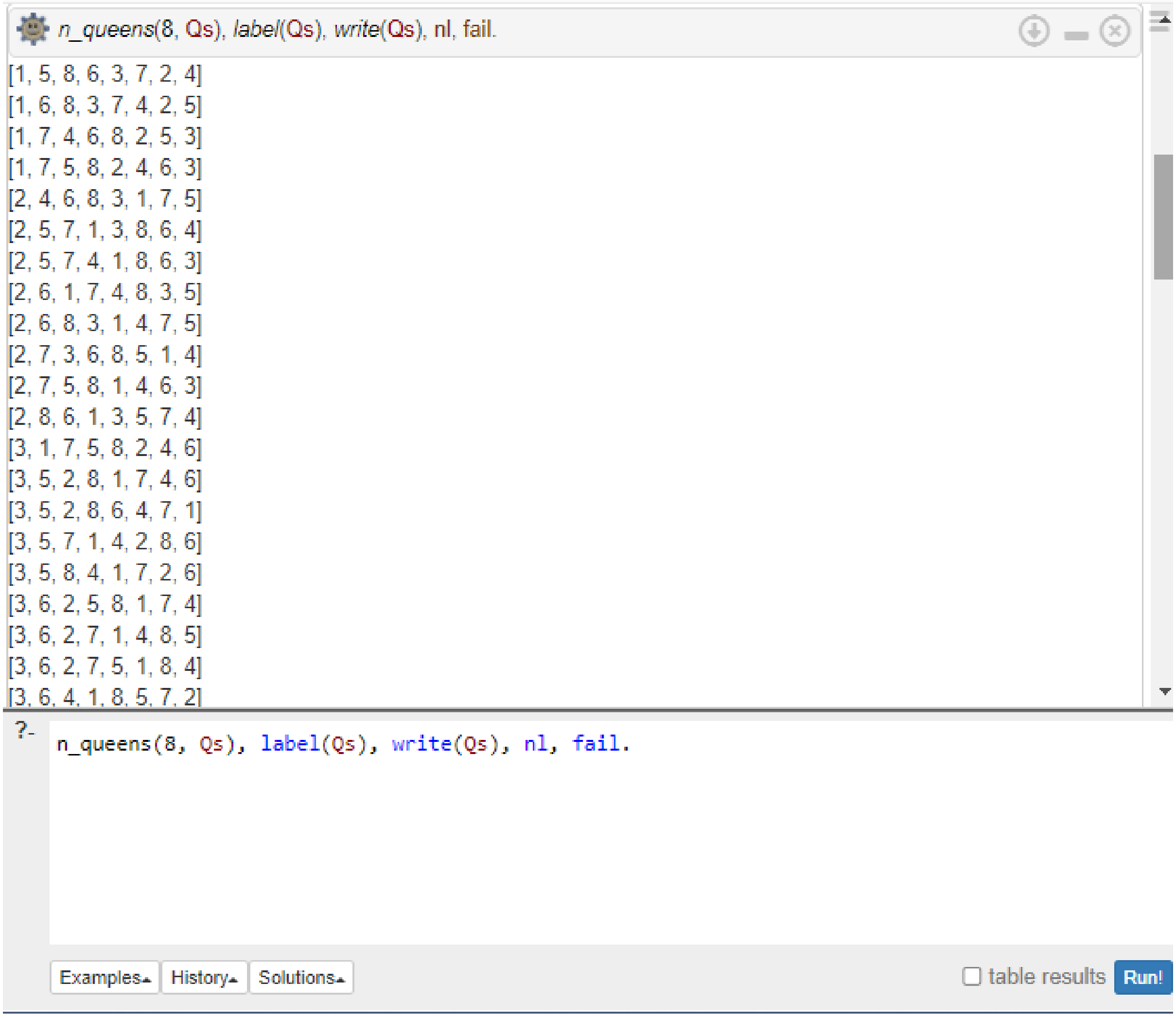
B) N-queen

Source code:



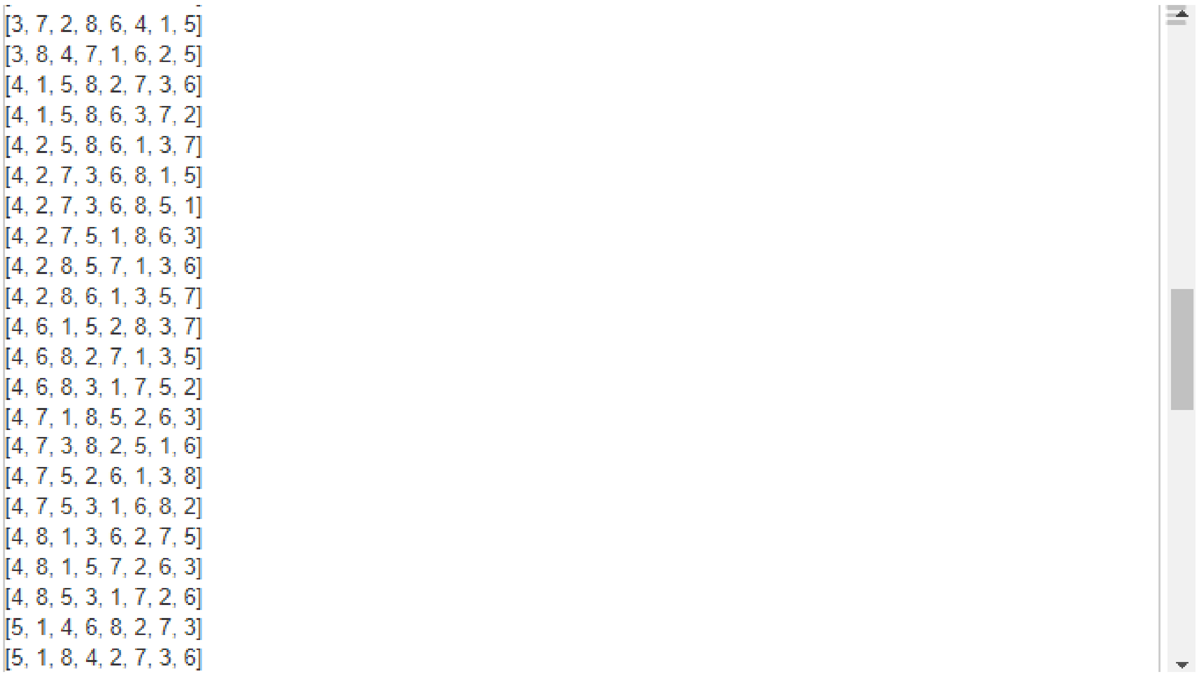
```
:- use_module(library(clpfd)).

% Step 1: Initialize N queens and insert in Qs
n_queens(N, Qs) :-
    length(Qs, N),
    Qs ins 1..N,
    safe_queens(Qs).

% Step 2: Set safe_queens to null
safe_queens([]).

% Step 3: Verify if attack is possible
safe_queens([Q|Qs]) :-
    safe_queens(Qs, Q, 1),
    safe_queens(Qs).

% Step 4: Continue till Qs id matches N
safe_queens([], _, _).

% Step 5: If Q meets no attack, declare Q as safe and add to safe_queens
safe_queens([Q|Qs], Q0, D0) :-
    Q0 #\= Q,
    abs(Q0 - Q) #\= D0,
    D1 #= D0 + 1,
    safe_queens(Qs, Q0, D1).

% Example usage:
% To solve N-Queens problem for N = 8
% Query: ?- n_queens(8, Qs), label(Qs), write(Qs), nl, fail.
% This will find all solutions for placing 8 queens on an 8x8 chessboard.
```

Output:

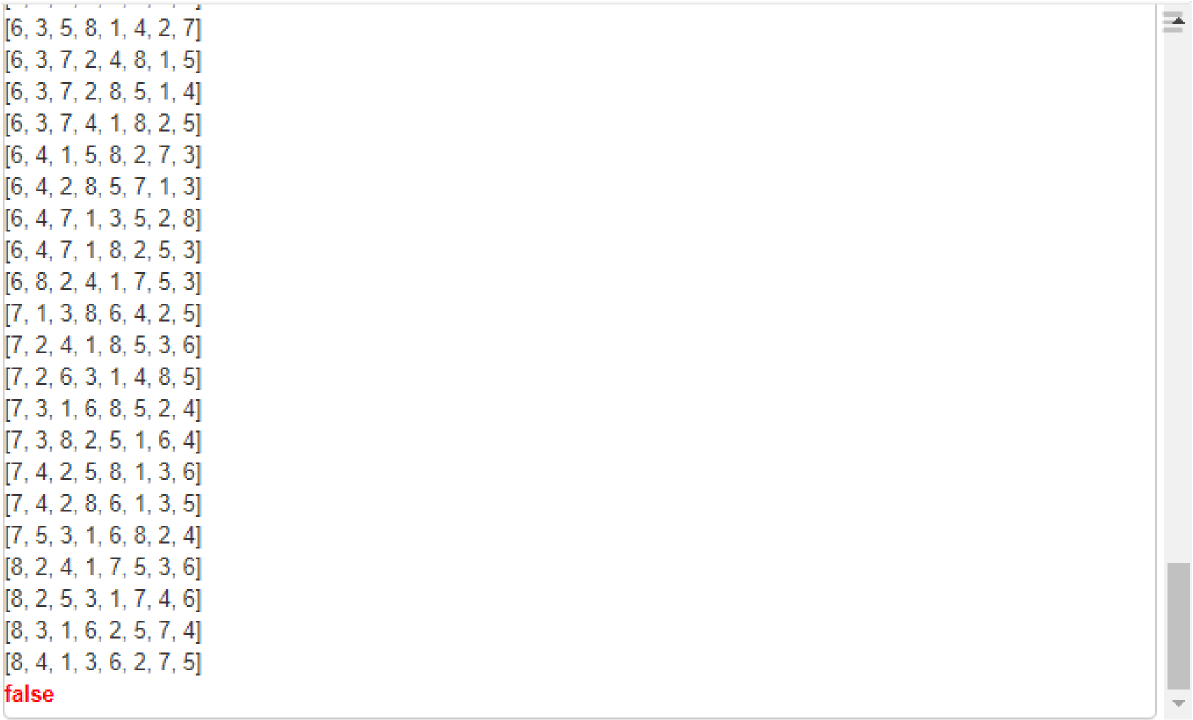


```
n_queens(8, Qs), label(Qs), write(Qs), nl, fail.
[1, 5, 8, 6, 3, 7, 2, 4]
[1, 6, 8, 3, 7, 4, 2, 5]
[1, 7, 4, 6, 8, 2, 5, 3]
[1, 7, 5, 8, 2, 4, 6, 3]
[2, 4, 6, 8, 3, 1, 7, 5]
[2, 5, 7, 1, 3, 8, 6, 4]
[2, 5, 7, 4, 1, 8, 6, 3]
[2, 6, 1, 7, 4, 8, 3, 5]
[2, 6, 8, 3, 1, 4, 7, 5]
[2, 7, 3, 6, 8, 5, 1, 4]
[2, 7, 5, 8, 1, 4, 6, 3]
[2, 8, 6, 1, 3, 5, 7, 4]
[3, 1, 7, 5, 8, 2, 4, 6]
[3, 5, 2, 8, 1, 7, 4, 6]
[3, 5, 2, 8, 6, 4, 7, 1]
[3, 5, 7, 1, 4, 2, 8, 6]
[3, 5, 8, 4, 1, 7, 2, 6]
[3, 6, 2, 5, 8, 1, 7, 4]
[3, 6, 2, 7, 1, 4, 8, 5]
[3, 6, 2, 7, 5, 1, 8, 4]
[3, 6, 4, 1, 8, 5, 7, 2]
```

```
?- n_queens(8, Qs), label(Qs), write(Qs), nl, fail.
```



```
[3, 7, 2, 8, 6, 4, 1, 5]
[3, 8, 4, 7, 1, 6, 2, 5]
[4, 1, 5, 8, 2, 7, 3, 6]
[4, 1, 5, 8, 6, 3, 7, 2]
[4, 2, 5, 8, 6, 1, 3, 7]
[4, 2, 7, 3, 6, 8, 1, 5]
[4, 2, 7, 3, 6, 8, 5, 1]
[4, 2, 7, 5, 1, 8, 6, 3]
[4, 2, 8, 5, 7, 1, 3, 6]
[4, 2, 8, 6, 1, 3, 5, 7]
[4, 6, 1, 5, 2, 8, 3, 7]
[4, 6, 8, 2, 7, 1, 3, 5]
[4, 6, 8, 3, 1, 7, 5, 2]
[4, 7, 1, 8, 5, 2, 6, 3]
[4, 7, 3, 8, 2, 5, 1, 6]
[4, 7, 5, 2, 6, 1, 3, 8]
[4, 7, 5, 3, 1, 6, 8, 2]
[4, 8, 1, 3, 6, 2, 7, 5]
[4, 8, 1, 5, 7, 2, 6, 3]
[4, 8, 5, 3, 1, 7, 2, 6]
[5, 1, 4, 6, 8, 2, 7, 3]
[5, 1, 8, 4, 2, 7, 3, 6]
```



```
[6, 3, 5, 8, 1, 4, 2, 7]
[6, 3, 7, 2, 4, 8, 1, 5]
[6, 3, 7, 2, 8, 5, 1, 4]
[6, 3, 7, 4, 1, 8, 2, 5]
[6, 4, 1, 5, 8, 2, 7, 3]
[6, 4, 2, 8, 5, 7, 1, 3]
[6, 4, 7, 1, 3, 5, 2, 8]
[6, 4, 7, 1, 8, 2, 5, 3]
[6, 8, 2, 4, 1, 7, 5, 3]
[7, 1, 3, 8, 6, 4, 2, 5]
[7, 2, 4, 1, 8, 5, 3, 6]
[7, 2, 6, 3, 1, 4, 8, 5]
[7, 3, 1, 6, 8, 5, 2, 4]
[7, 3, 8, 2, 5, 1, 6, 4]
[7, 4, 2, 5, 8, 1, 3, 6]
[7, 4, 2, 8, 6, 1, 3, 5]
[7, 5, 3, 1, 6, 8, 2, 4]
[8, 2, 4, 1, 7, 5, 3, 6]
[8, 2, 5, 3, 1, 7, 4, 6]
[8, 3, 1, 6, 2, 5, 7, 4]
[8, 4, 1, 3, 6, 2, 7, 5]
false
```