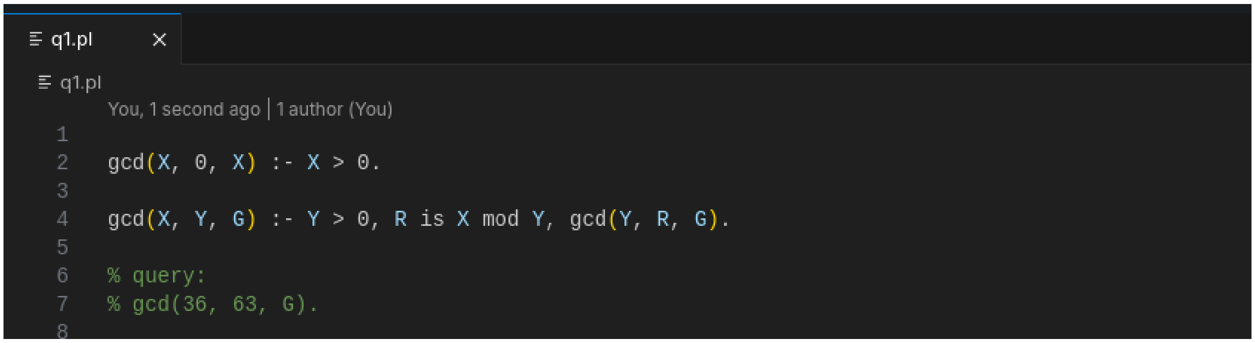
CSE 354N LAB 3

Arnav Jain - 220002018

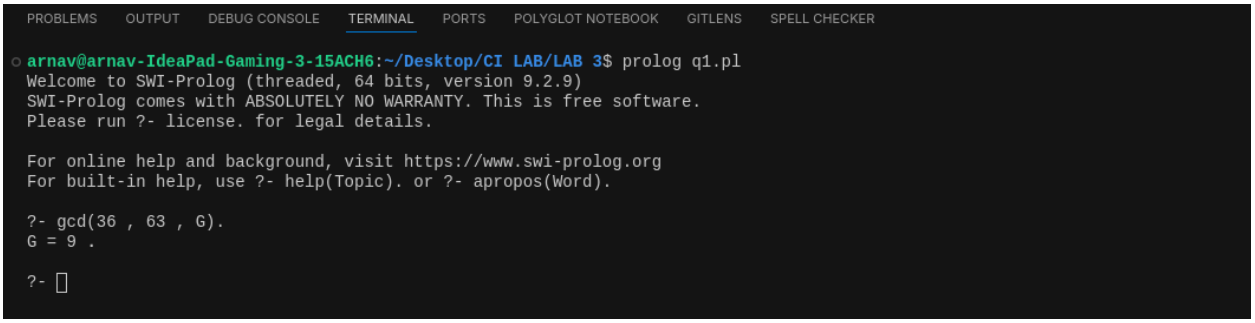
Question 1

Code:



```prolog

gcd(X, 0, X) :- X > 0.

gcd(X, Y, G) :- Y > 0, R is X mod Y, gcd(Y, R, G).

% query:
% gcd(36, 63, G).

```

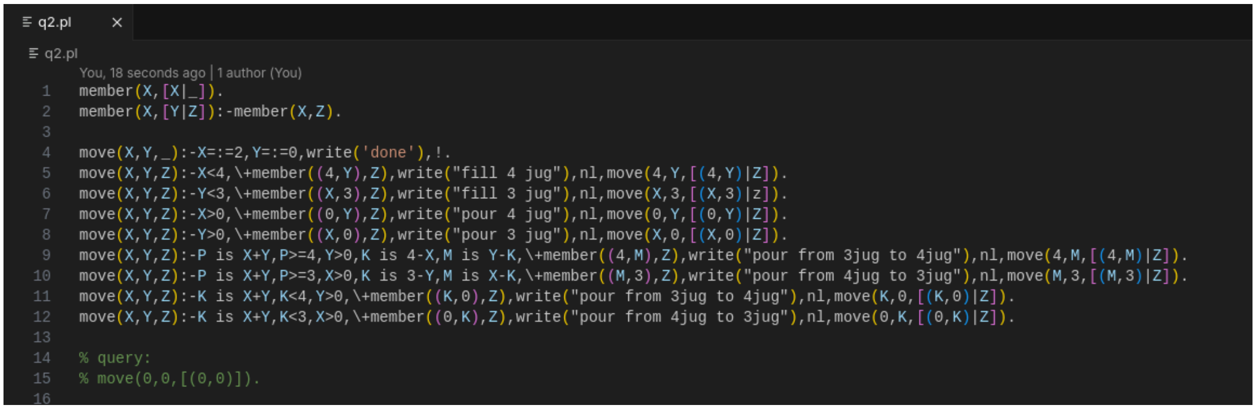
Result:



```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/CI LAB/LAB 3$ prolog q1.pl
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).

?- gcd(36 , 63 , G).
G = 9 .

?- 
```

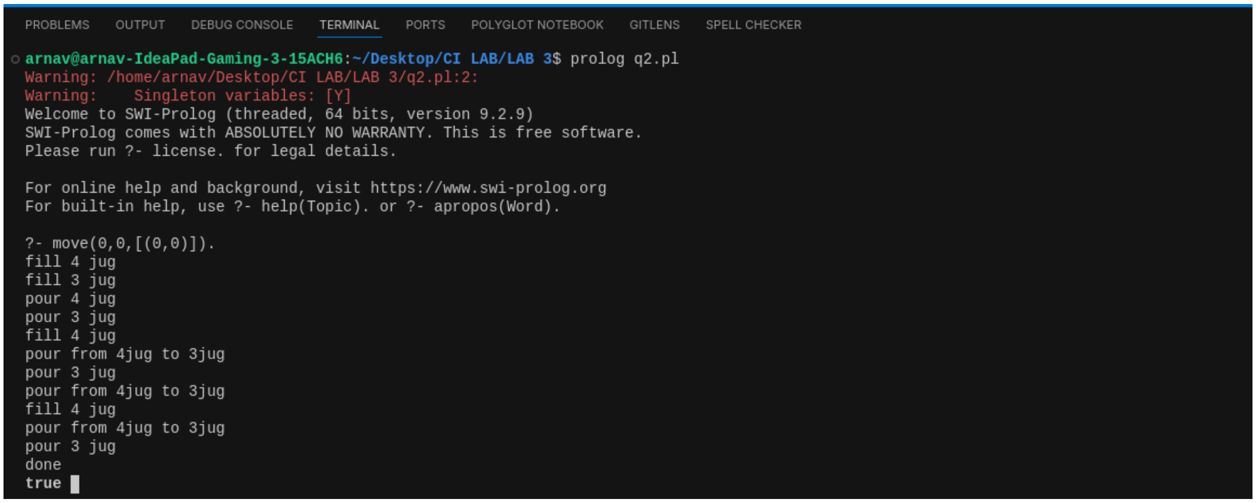
Question 2

Code:



```prolog
member(X,[X|_]).
member(X,[Y|Z]):-member(X,Z).

move(X,Y,_):-X=:=2,Y=:=0,write('done'),!.
move(X,Y,Z):-X<4,\+member((4,Y),Z),write("fill 4 jug"),nl,move(4,Y,[(4,Y)|Z]).
move(X,Y,Z):-Y<3,\+member((X,3),Z),write("fill 3 jug"),nl,move(X,3,[(X,3)|z]).
move(X,Y,Z):-X>0,\+member((0,Y),Z),write("pour 4 jug"),nl,move(0,Y,[(0,Y)|Z]).
move(X,Y,Z):-Y>0,\+member((X,0),Z),write("pour 3 jug"),nl,move(X,0,[(X,0)|Z]).
move(X,Y,Z):-P is X+Y,P>=4,Y>0,K is 4-X,M is Y-K,\+member((4,M),Z),write("pour from 3jug to 4jug"),nl,move(4,M,[(4,M)|Z]).
move(X,Y,Z):-P is X+Y,P>=3,X>0,K is 3-Y,M is X-K,\+member((M,3),Z),write("pour from 4jug to 3jug"),nl,move(M,3,[(M,3)|Z]).
move(X,Y,Z):-K is X+Y,K<4,Y>0,\+member((K,0),Z),write("pour from 3jug to 4jug"),nl,move(K,0,[(K,0)|Z]).
move(X,Y,Z):-K is X+Y,K<3,X>0,\+member((0,K),Z),write("pour from 4jug to 3jug"),nl,move(0,K,[(0,K)|Z]).

% query:
% move(0,0,[(0,0)]).
```

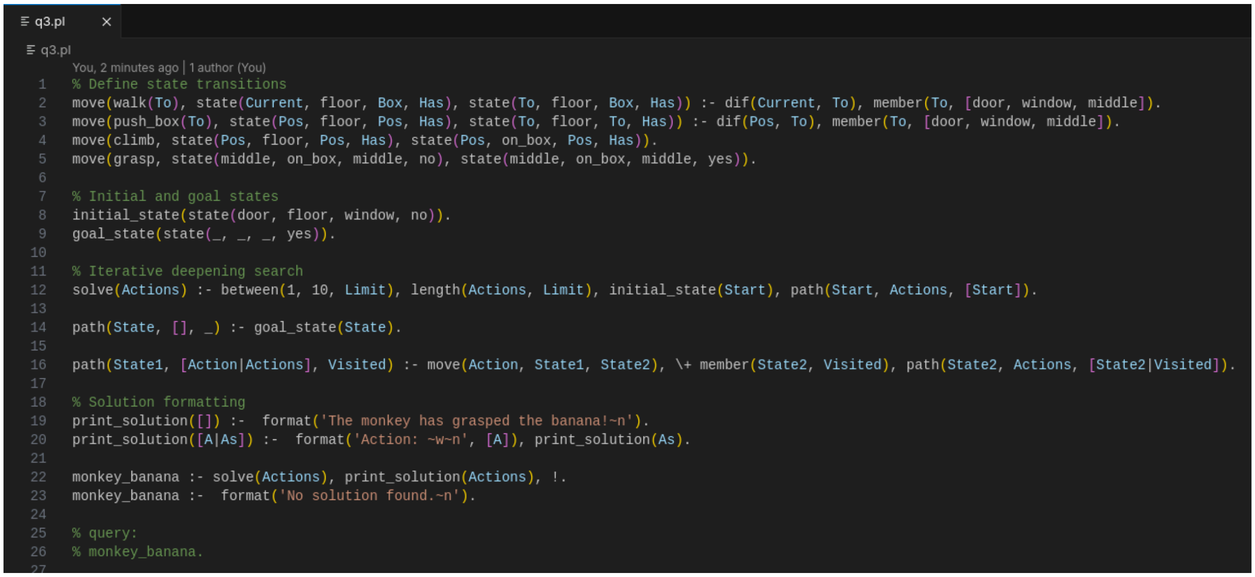
Result:



```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/CI LAB/LAB 3$ prolog q2.pl
Warning: /home/arnav/Desktop/CI LAB/LAB 3/q2.pl:2:
Warning:    Singleton variables: [Y]
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).

?- move(0,0,[(0,0)]).
fill 4 jug
fill 3 jug
pour 4 jug
pour 3 jug
fill 4 jug
pour from 4jug to 3jug
pour 3 jug
pour from 4jug to 3jug
fill 4 jug
pour from 4jug to 3jug
pour 3 jug
done
true
```

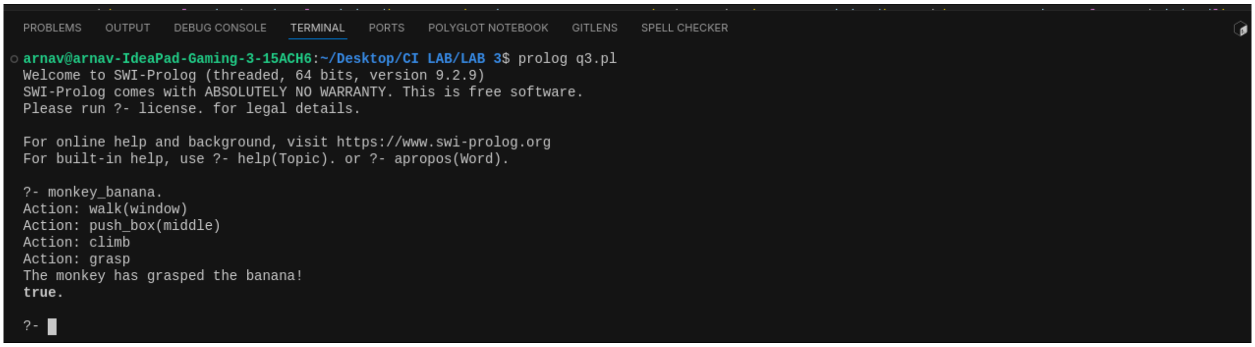
Question 3

Code:



```prolog
% Define state transitions
move(walk(To), state(Current, floor, Box, Has), state(To, floor, Box, Has)) :- dif(Current, To), member(To, [door, window, middle]).
move(push_box(To), state(Pos, floor, Pos, Has), state(To, floor, To, Has)) :- dif(Pos, To), member(To, [door, window, middle]).
move(climb, state(Pos, floor, Pos, Has), state(Pos, on_box, Pos, Has)).
move(grasp, state(middle, on_box, middle, no), state(middle, on_box, middle, yes)).

% Initial and goal states
initial_state(state(door, floor, window, no)).
goal_state(state(_, _, _, yes)).

% Iterative deepening search
solve(Actions) :- between(1, 10, Limit), length(Actions, Limit), initial_state(Start), path(Start, Actions, [Start]).

path(State, [], _) :- goal_state(State).

path(State1, [Action|Actions], Visited) :- move(Action, State1, State2), \+ member(State2, Visited), path(State2, Actions, [State2|Visited]).

% Solution formatting
print_solution([]) :-  format('The monkey has grasped the banana!~n').
print_solution([A|As]) :-  format('Action: ~w~n', [A]), print_solution(As).

monkey_banana :- solve(Actions), print_solution(Actions), !.
monkey_banana :-  format('No solution found.~n').

% query:
% monkey_banana.
```

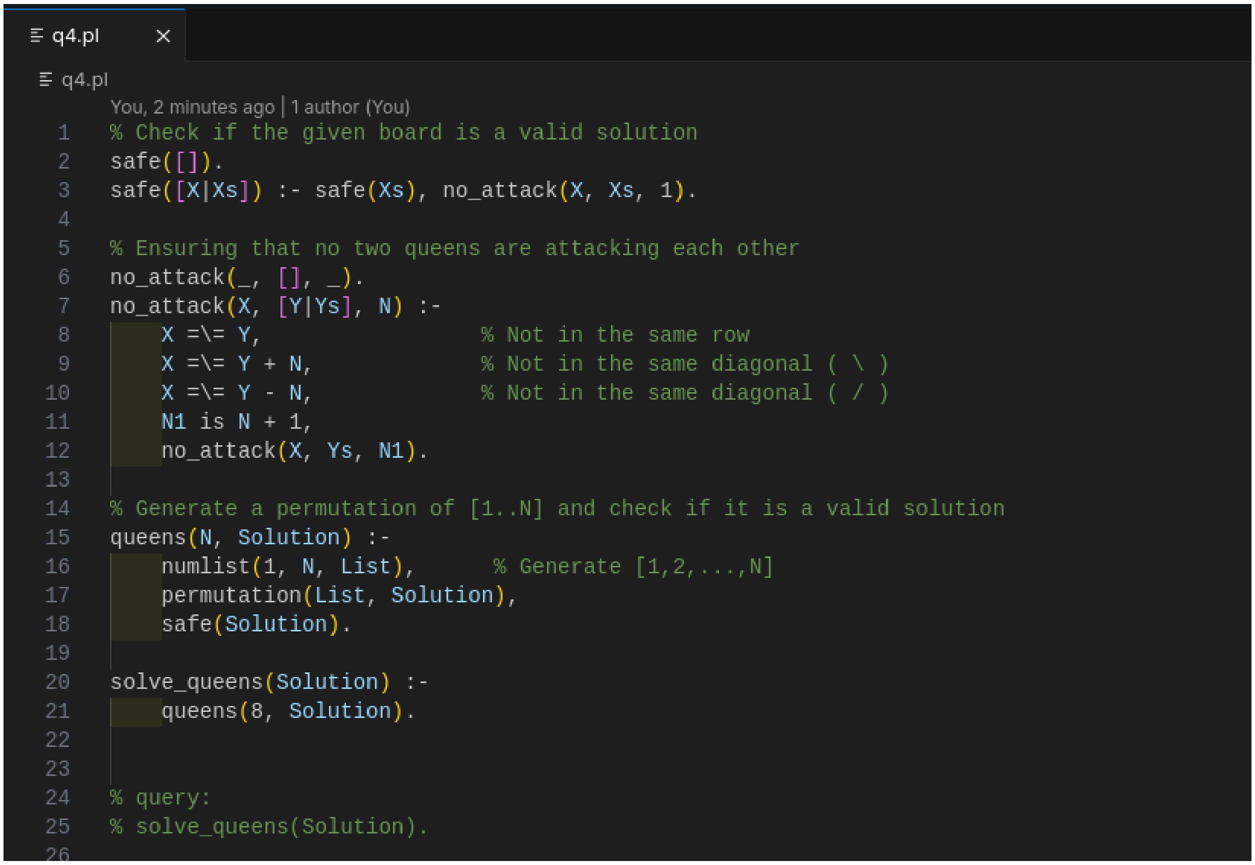
Result:



```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/CI LAB/LAB 3$ prolog q3.pl
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).

?- monkey_banana.
Action: walk(window)
Action: push_box(middle)
Action: climb
Action: grasp
The monkey has grasped the banana!
true.

?- 
```

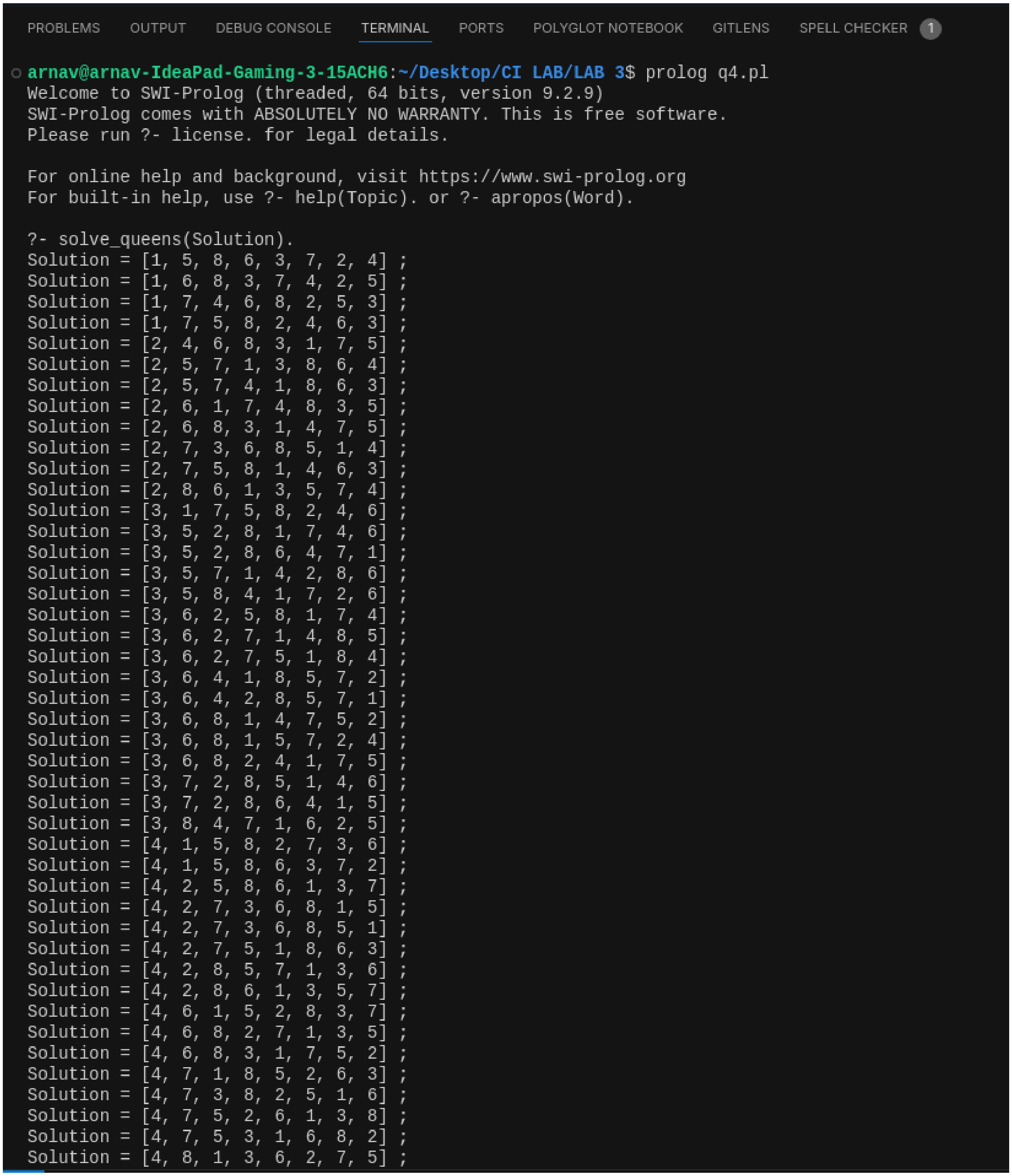
Question 4

Code:

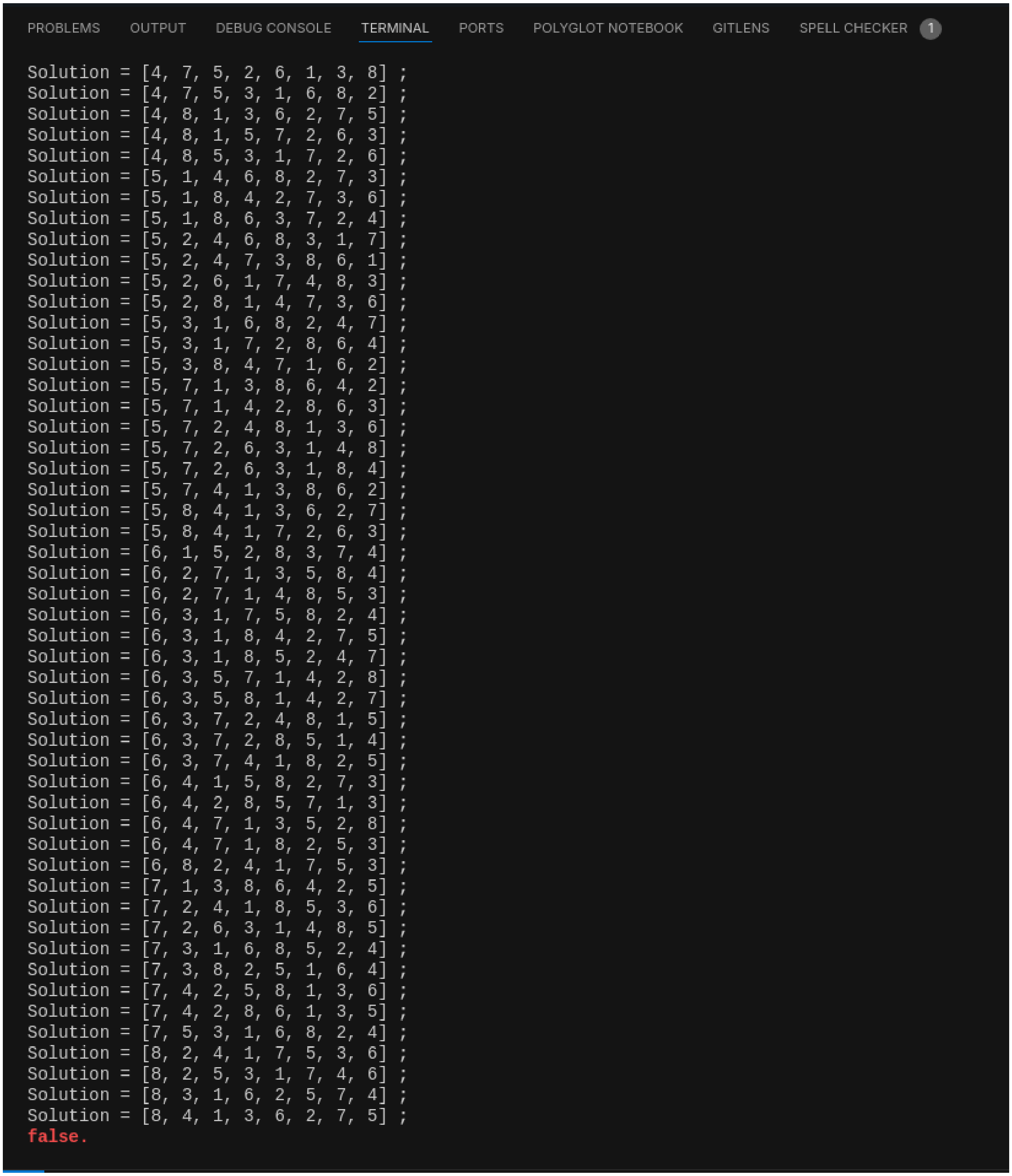


```prolog
% Check if the given board is a valid solution
safe([]).
safe([X|Xs]) :- safe(Xs), no_attack(X, Xs, 1).

% Ensuring that no two queens are attacking each other
no_attack(_, [], _).
no_attack(X, [Y|Ys], N) :-
    X =\= Y,                 % Not in the same row
    X =\= Y + N,             % Not in the same diagonal ( \ )
    X =\= Y - N,             % Not in the same diagonal ( / )
    N1 is N + 1,
    no_attack(X, Ys, N1).

% Generate a permutation of [1..N] and check if it is a valid solution
queens(N, Solution) :-
    numlist(1, N, List),      % Generate [1,2,...,N]
    permutation(List, Solution),
    safe(Solution).

solve_queens(Solution) :-
    queens(8, Solution).


% query:
% solve_queens(Solution).
```

Result:



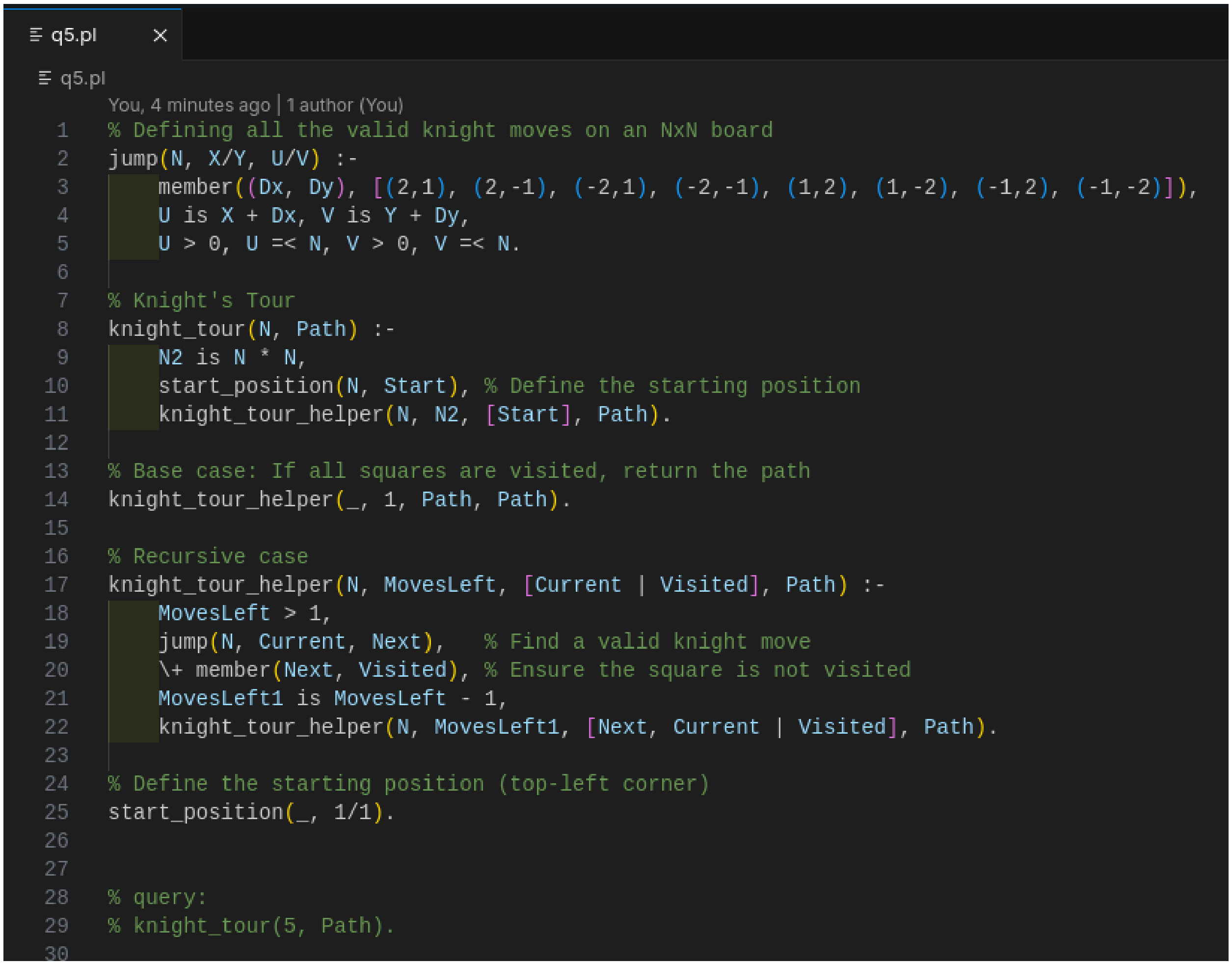
```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/CI LAB/LAB 3$ prolog q4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- solve_queens(Solution).
Solution = [1, 5, 8, 6, 3, 7, 2, 4] ;
Solution = [1, 6, 8, 3, 7, 4, 2, 5] ;
Solution = [1, 7, 4, 6, 8, 2, 5, 3] ;
Solution = [1, 7, 5, 8, 2, 4, 6, 3] ;
Solution = [2, 4, 6, 8, 3, 1, 7, 5] ;
Solution = [2, 5, 7, 1, 3, 8, 6, 4] ;
Solution = [2, 5, 7, 4, 1, 8, 6, 3] ;
Solution = [2, 6, 1, 7, 4, 8, 3, 5] ;
Solution = [2, 6, 8, 3, 1, 4, 7, 5] ;
Solution = [2, 7, 3, 6, 8, 5, 1, 4] ;
Solution = [2, 7, 5, 8, 1, 4, 6, 3] ;
Solution = [2, 8, 6, 1, 3, 5, 7, 4] ;
Solution = [3, 1, 7, 5, 8, 2, 4, 6] ;
Solution = [3, 5, 2, 8, 1, 7, 4, 6] ;
Solution = [3, 5, 2, 8, 6, 4, 7, 1] ;
Solution = [3, 5, 7, 1, 4, 2, 8, 6] ;
Solution = [3, 5, 8, 4, 1, 7, 2, 6] ;
Solution = [3, 6, 2, 5, 8, 1, 7, 4] ;
Solution = [3, 6, 2, 7, 1, 4, 8, 5] ;
Solution = [3, 6, 2, 7, 5, 1, 8, 4] ;
Solution = [3, 6, 4, 1, 8, 5, 7, 2] ;
Solution = [3, 6, 4, 2, 8, 5, 7, 1] ;
Solution = [3, 6, 8, 1, 4, 7, 5, 2] ;
Solution = [3, 6, 8, 1, 5, 7, 2, 4] ;
Solution = [3, 6, 8, 2, 4, 1, 7, 5] ;
Solution = [3, 7, 2, 8, 5, 1, 4, 6] ;
Solution = [3, 7, 2, 8, 6, 4, 1, 5] ;
Solution = [3, 8, 4, 7, 1, 6, 2, 5] ;
Solution = [4, 1, 5, 8, 2, 7, 3, 6] ;
Solution = [4, 1, 5, 8, 6, 3, 7, 2] ;
Solution = [4, 2, 5, 8, 6, 1, 3, 7] ;
Solution = [4, 2, 7, 3, 6, 8, 1, 5] ;
Solution = [4, 2, 7, 3, 6, 8, 5, 1] ;
Solution = [4, 2, 7, 5, 1, 8, 6, 3] ;
Solution = [4, 2, 8, 5, 7, 1, 3, 6] ;
Solution = [4, 2, 8, 6, 1, 3, 5, 7] ;
Solution = [4, 6, 1, 5, 2, 8, 3, 7] ;
Solution = [4, 6, 8, 2, 7, 1, 3, 5] ;
Solution = [4, 6, 8, 3, 1, 7, 5, 2] ;
Solution = [4, 7, 1, 8, 5, 2, 6, 3] ;
Solution = [4, 7, 3, 8, 2, 5, 1, 6] ;
Solution = [4, 7, 5, 2, 6, 1, 3, 8] ;
Solution = [4, 7, 5, 3, 1, 6, 8, 2] ;
Solution = [4, 8, 1, 3, 6, 2, 7, 5] ;
```



```
Solution = [4, 7, 5, 2, 6, 1, 3, 8] ;
Solution = [4, 7, 5, 3, 1, 6, 8, 2] ;
Solution = [4, 8, 1, 3, 6, 2, 7, 5] ;
Solution = [4, 8, 1, 5, 7, 2, 6, 3] ;
Solution = [4, 8, 5, 3, 1, 7, 2, 6] ;
Solution = [5, 1, 4, 6, 8, 2, 7, 3] ;
Solution = [5, 1, 8, 4, 2, 7, 3, 6] ;
Solution = [5, 1, 8, 6, 3, 7, 2, 4] ;
Solution = [5, 2, 4, 6, 8, 3, 1, 7] ;
Solution = [5, 2, 4, 7, 3, 8, 6, 1] ;
Solution = [5, 2, 6, 1, 7, 4, 8, 3] ;
Solution = [5, 2, 8, 1, 4, 7, 3, 6] ;
Solution = [5, 3, 1, 6, 8, 2, 4, 7] ;
Solution = [5, 3, 1, 7, 2, 8, 6, 4] ;
Solution = [5, 3, 8, 4, 7, 1, 6, 2] ;
Solution = [5, 7, 1, 3, 8, 6, 4, 2] ;
Solution = [5, 7, 1, 4, 2, 8, 6, 3] ;
Solution = [5, 7, 2, 4, 8, 1, 3, 6] ;
Solution = [5, 7, 2, 6, 3, 1, 4, 8] ;
Solution = [5, 7, 2, 6, 3, 1, 8, 4] ;
Solution = [5, 7, 4, 1, 3, 8, 6, 2] ;
Solution = [5, 8, 4, 1, 3, 6, 2, 7] ;
Solution = [5, 8, 4, 1, 7, 2, 6, 3] ;
Solution = [6, 1, 5, 2, 8, 3, 7, 4] ;
Solution = [6, 2, 7, 1, 3, 5, 8, 4] ;
Solution = [6, 2, 7, 1, 4, 8, 5, 3] ;
Solution = [6, 3, 1, 7, 5, 8, 2, 4] ;
Solution = [6, 3, 1, 8, 4, 2, 7, 5] ;
Solution = [6, 3, 1, 8, 5, 2, 4, 7] ;
Solution = [6, 3, 5, 7, 1, 4, 2, 8] ;
Solution = [6, 3, 5, 8, 1, 4, 2, 7] ;
Solution = [6, 3, 7, 2, 4, 8, 1, 5] ;
Solution = [6, 3, 7, 2, 8, 5, 1, 4] ;
Solution = [6, 3, 7, 4, 1, 8, 2, 5] ;
Solution = [6, 4, 1, 5, 8, 2, 7, 3] ;
Solution = [6, 4, 2, 8, 5, 7, 1, 3] ;
Solution = [6, 4, 7, 1, 3, 5, 2, 8] ;
Solution = [6, 4, 7, 1, 8, 2, 5, 3] ;
Solution = [6, 8, 2, 4, 1, 7, 5, 3] ;
Solution = [7, 1, 3, 8, 6, 4, 2, 5] ;
Solution = [7, 2, 4, 1, 8, 5, 3, 6] ;
Solution = [7, 2, 6, 3, 1, 4, 8, 5] ;
Solution = [7, 3, 1, 6, 8, 5, 2, 4] ;
Solution = [7, 3, 8, 2, 5, 1, 6, 4] ;
Solution = [7, 4, 2, 5, 8, 1, 3, 6] ;
Solution = [7, 4, 2, 8, 6, 1, 3, 5] ;
Solution = [7, 5, 3, 1, 6, 8, 2, 4] ;
Solution = [8, 2, 4, 1, 7, 5, 3, 6] ;
Solution = [8, 2, 5, 3, 1, 7, 4, 6] ;
Solution = [8, 3, 1, 6, 2, 5, 7, 4] ;
Solution = [8, 4, 1, 3, 6, 2, 7, 5] ;
false.
```

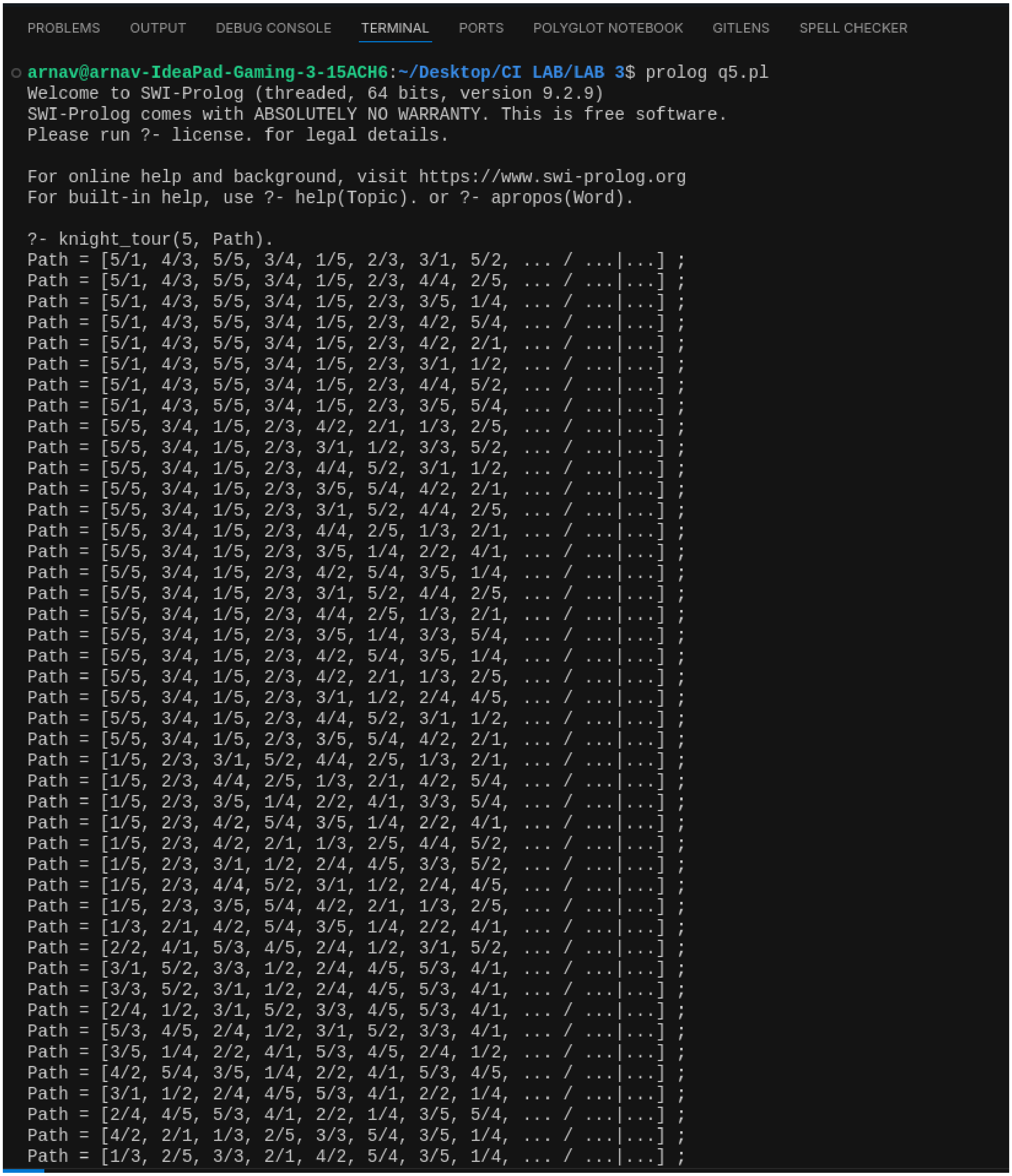
Question 5

Code:

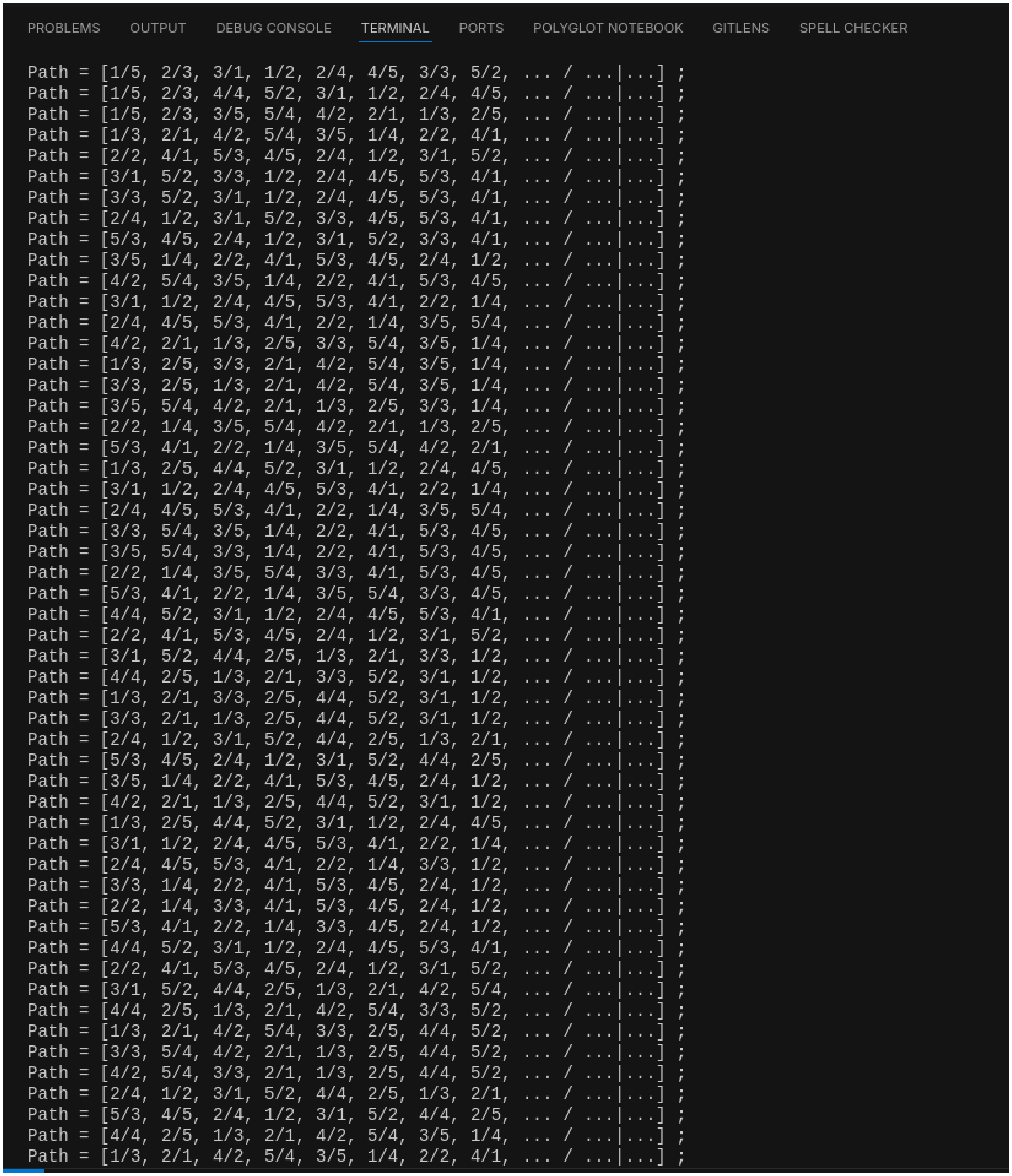


```
% Defining all the valid knight moves on an NxN board
jump(N, X/Y, U/V) :-
    member((Dx, Dy), [(2,1), (2,-1), (-2,1), (-2,-1), (1,2), (1,-2), (-1,2), (-1,-2)]),
    U is X + Dx, V is Y + Dy,
    U > 0, U =< N, V > 0, V =< N.

% Knight's Tour
knight_tour(N, Path) :-
    N2 is N * N,
    start_position(N, Start), % Define the starting position
    knight_tour_helper(N, N2, [Start], Path).

% Base case: If all squares are visited, return the path
knight_tour_helper(_, 1, Path, Path).

% Recursive case
knight_tour_helper(N, MovesLeft, [Current | Visited], Path) :-
    MovesLeft > 1,
    jump(N, Current, Next),    % Find a valid knight move
    \+ member(Next, Visited), % Ensure the square is not visited
    MovesLeft1 is MovesLeft - 1,
    knight_tour_helper(N, MovesLeft1, [Next, Current | Visited], Path).

% Define the starting position (top-left corner)
start_position(_, 1/1).


% query:
% knight_tour(5, Path).
```

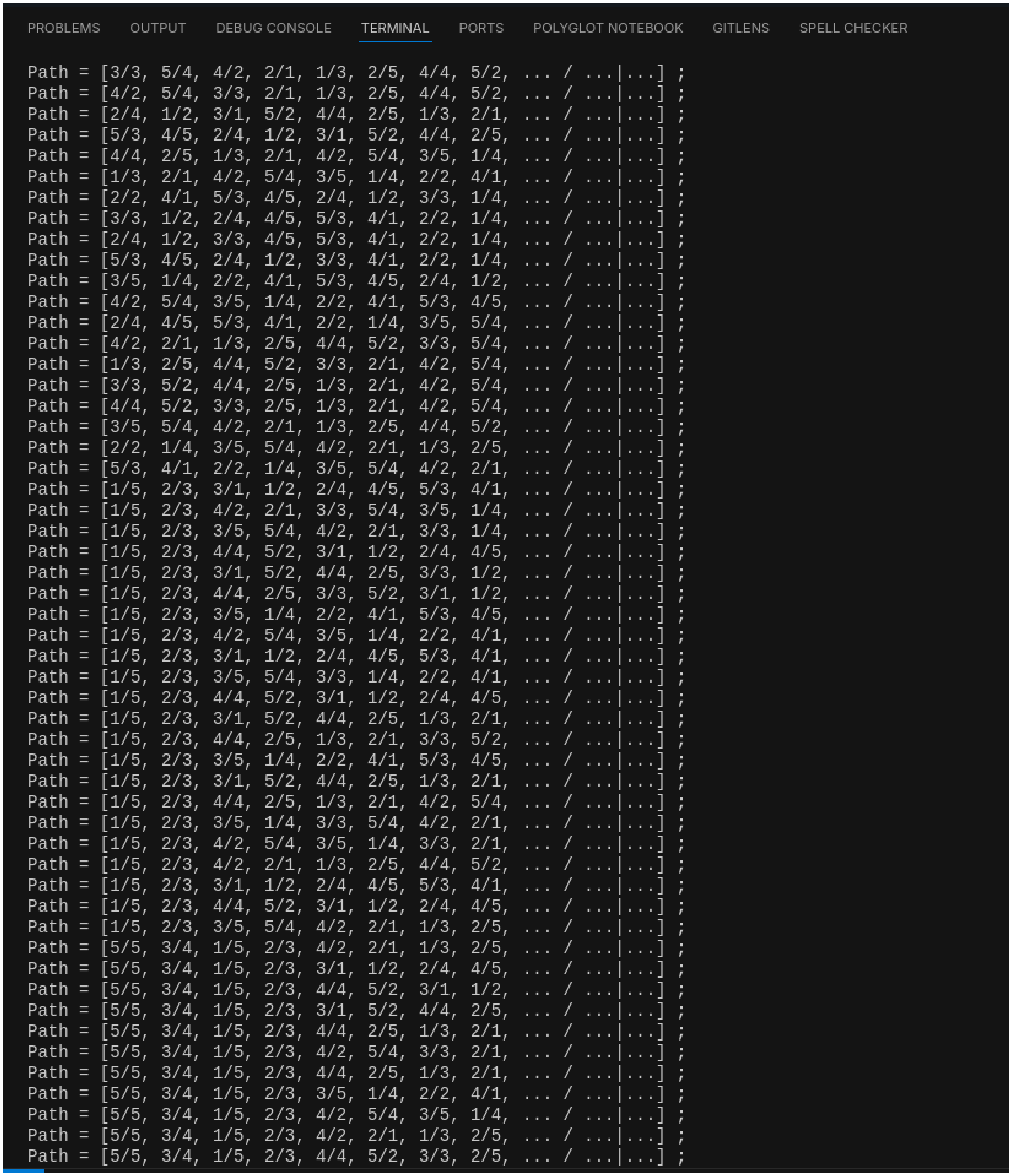
Result:



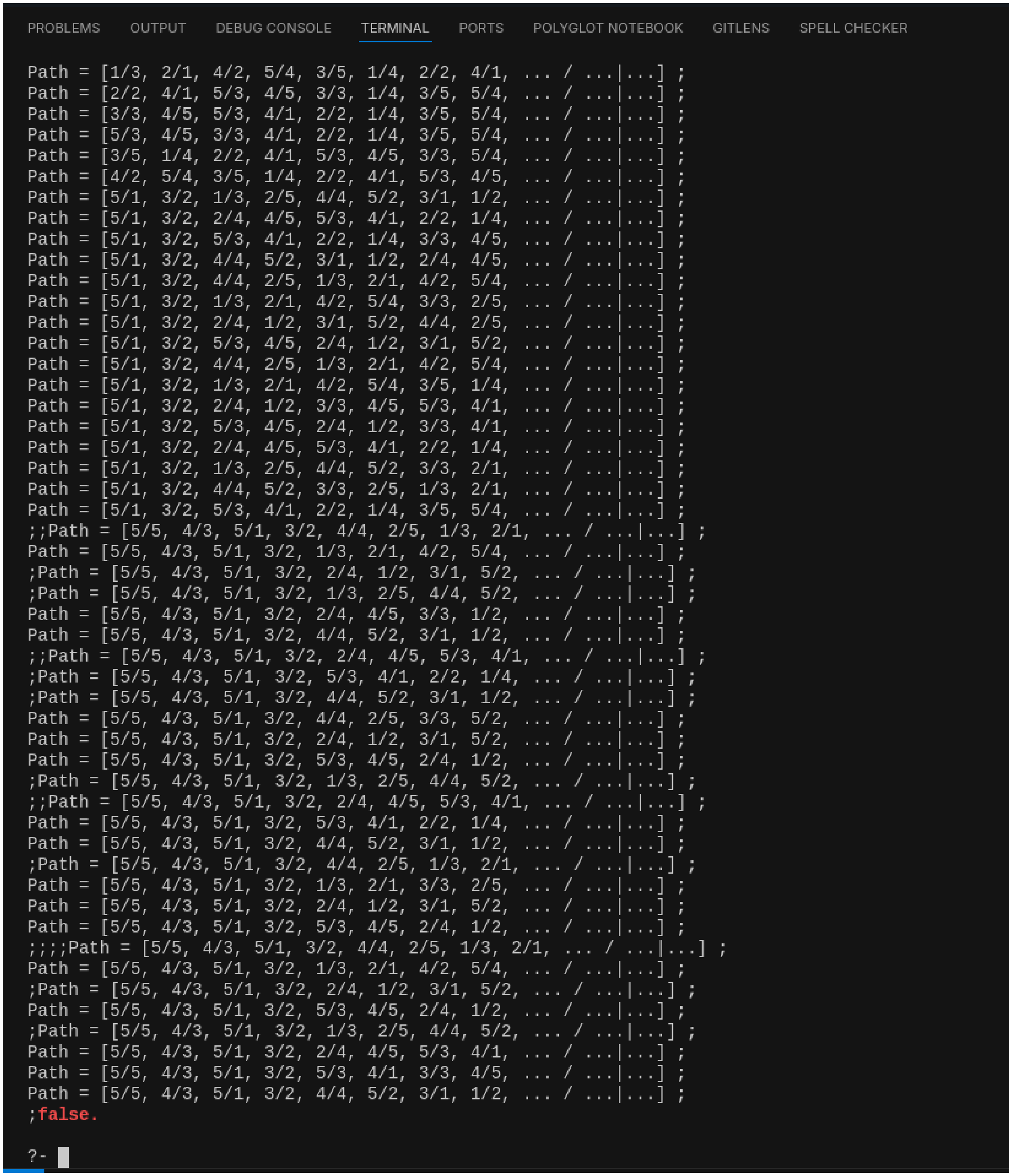
```
○ arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/CI LAB/LAB 3$ prolog q5.pl
  Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
  SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
  Please run ?- license. for legal details.

  For online help and background, visit https://www.swi-prolog.org
  For built-in help, use ?- help(Topic). or ?- apropos(Word).

  ?- knight_tour(5, Path).
  Path = [5/1, 4/3, 5/5, 3/4, 1/5, 2/3, 3/1, 5/2, ... / ...|...] ;
  Path = [5/1, 4/3, 5/5, 3/4, 1/5, 2/3, 4/4, 2/5, ... / ...|...] ;
  Path = [5/1, 4/3, 5/5, 3/4, 1/5, 2/3, 3/5, 1/4, ... / ...|...] ;
  Path = [5/1, 4/3, 5/5, 3/4, 1/5, 2/3, 4/2, 5/4, ... / ...|...] ;
  Path = [5/1, 4/3, 5/5, 3/4, 1/5, 2/3, 4/2, 2/1, ... / ...|...] ;
  Path = [5/1, 4/3, 5/5, 3/4, 1/5, 2/3, 3/1, 1/2, ... / ...|...] ;
  Path = [5/1, 4/3, 5/5, 3/4, 1/5, 2/3, 4/4, 5/2, ... / ...|...] ;
  Path = [5/1, 4/3, 5/5, 3/4, 1/5, 2/3, 3/5, 5/4, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 4/2, 2/1, 1/3, 2/5, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 3/1, 1/2, 3/3, 5/2, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 3/5, 5/4, 4/2, 2/1, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 3/1, 5/2, 4/4, 2/5, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 3/5, 1/4, 2/2, 4/1, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 4/2, 5/4, 3/5, 1/4, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 3/1, 5/2, 4/4, 2/5, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 3/5, 1/4, 3/3, 5/4, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 4/2, 5/4, 3/5, 1/4, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 4/2, 2/1, 1/3, 2/5, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 3/1, 1/2, 2/4, 4/5, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
  Path = [5/5, 3/4, 1/5, 2/3, 3/5, 5/4, 4/2, 2/1, ... / ...|...] ;
  Path = [1/5, 2/3, 3/1, 5/2, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
  Path = [1/5, 2/3, 4/4, 2/5, 1/3, 2/1, 4/2, 5/4, ... / ...|...] ;
  Path = [1/5, 2/3, 3/5, 1/4, 2/2, 4/1, 3/3, 5/4, ... / ...|...] ;
  Path = [1/5, 2/3, 4/2, 5/4, 3/5, 1/4, 2/2, 4/1, ... / ...|...] ;
  Path = [1/5, 2/3, 4/2, 2/1, 1/3, 2/5, 4/4, 5/2, ... / ...|...] ;
  Path = [1/5, 2/3, 3/1, 1/2, 2/4, 4/5, 3/3, 5/2, ... / ...|...] ;
  Path = [1/5, 2/3, 4/4, 5/2, 3/1, 1/2, 2/4, 4/5, ... / ...|...] ;
  Path = [1/5, 2/3, 3/5, 5/4, 4/2, 2/1, 1/3, 2/5, ... / ...|...] ;
  Path = [1/3, 2/1, 4/2, 5/4, 3/5, 1/4, 2/2, 4/1, ... / ...|...] ;
  Path = [2/2, 4/1, 5/3, 4/5, 2/4, 1/2, 3/1, 5/2, ... / ...|...] ;
  Path = [3/1, 5/2, 3/3, 1/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
  Path = [3/3, 5/2, 3/1, 1/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
  Path = [2/4, 1/2, 3/1, 5/2, 3/3, 4/5, 5/3, 4/1, ... / ...|...] ;
  Path = [5/3, 4/5, 2/4, 1/2, 3/1, 5/2, 3/3, 4/1, ... / ...|...] ;
  Path = [3/5, 1/4, 2/2, 4/1, 5/3, 4/5, 2/4, 1/2, ... / ...|...] ;
  Path = [4/2, 5/4, 3/5, 1/4, 2/2, 4/1, 5/3, 4/5, ... / ...|...] ;
  Path = [3/1, 1/2, 2/4, 4/5, 5/3, 4/1, 2/2, 1/4, ... / ...|...] ;
  Path = [2/4, 4/5, 5/3, 4/1, 2/2, 1/4, 3/5, 5/4, ... / ...|...] ;
  Path = [4/2, 2/1, 1/3, 2/5, 3/3, 5/4, 3/5, 1/4, ... / ...|...] ;
  Path = [1/3, 2/5, 3/3, 2/1, 4/2, 5/4, 3/5, 1/4, ... / ...|...] ;
```



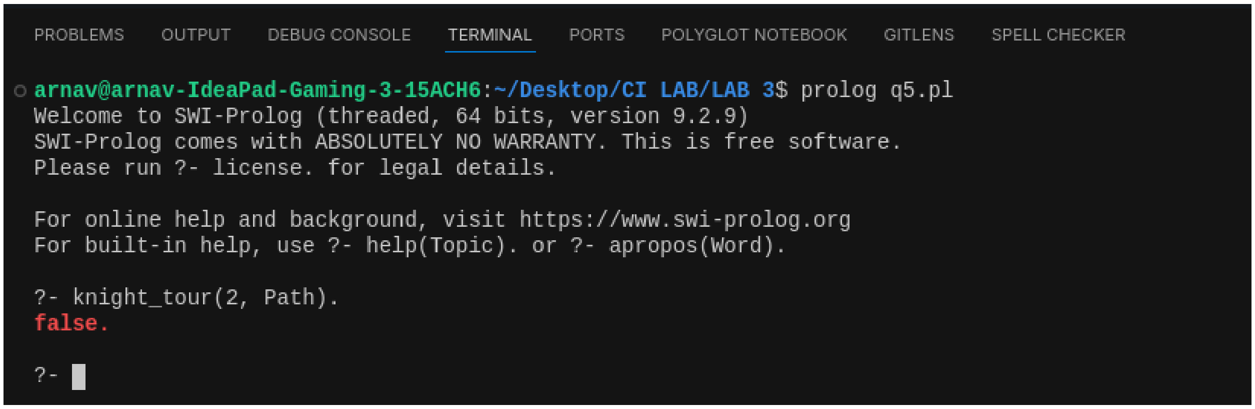
```
Path = [1/5, 2/3, 3/1, 1/2, 2/4, 4/5, 3/3, 5/2, ... / ...|...] ;
Path = [1/5, 2/3, 4/4, 5/2, 3/1, 1/2, 2/4, 4/5, ... / ...|...] ;
Path = [1/5, 2/3, 3/5, 5/4, 4/2, 2/1, 1/3, 2/5, ... / ...|...] ;
Path = [1/3, 2/1, 4/2, 5/4, 3/5, 1/4, 2/2, 4/1, ... / ...|...] ;
Path = [2/2, 4/1, 5/3, 4/5, 2/4, 1/2, 3/1, 5/2, ... / ...|...] ;
Path = [3/1, 5/2, 3/3, 1/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [3/3, 5/2, 3/1, 1/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [2/4, 1/2, 3/1, 5/2, 3/3, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [5/3, 4/5, 2/4, 1/2, 3/1, 5/2, 3/3, 4/1, ... / ...|...] ;
Path = [3/5, 1/4, 2/2, 4/1, 5/3, 4/5, 2/4, 1/2, ... / ...|...] ;
Path = [4/2, 5/4, 3/5, 1/4, 2/2, 4/1, 5/3, 4/5, ... / ...|...] ;
Path = [3/1, 1/2, 2/4, 4/5, 5/3, 4/1, 2/2, 1/4, ... / ...|...] ;
Path = [2/4, 4/5, 5/3, 4/1, 2/2, 1/4, 3/5, 5/4, ... / ...|...] ;
Path = [4/2, 2/1, 1/3, 2/5, 3/3, 5/4, 3/5, 1/4, ... / ...|...] ;
Path = [1/3, 2/5, 3/3, 2/1, 4/2, 5/4, 3/5, 1/4, ... / ...|...] ;
Path = [3/3, 2/5, 1/3, 2/1, 4/2, 5/4, 3/5, 1/4, ... / ...|...] ;
Path = [3/5, 5/4, 4/2, 2/1, 1/3, 2/5, 3/3, 1/4, ... / ...|...] ;
Path = [2/2, 1/4, 3/5, 5/4, 4/2, 2/1, 1/3, 2/5, ... / ...|...] ;
Path = [5/3, 4/1, 2/2, 1/4, 3/5, 5/4, 4/2, 2/1, ... / ...|...] ;
Path = [1/3, 2/5, 4/4, 5/2, 3/1, 1/2, 2/4, 4/5, ... / ...|...] ;
Path = [3/1, 1/2, 2/4, 4/5, 5/3, 4/1, 2/2, 1/4, ... / ...|...] ;
Path = [2/4, 4/5, 5/3, 4/1, 2/2, 1/4, 3/5, 5/4, ... / ...|...] ;
Path = [3/3, 5/4, 3/5, 1/4, 2/2, 4/1, 5/3, 4/5, ... / ...|...] ;
Path = [3/5, 5/4, 3/3, 1/4, 2/2, 4/1, 5/3, 4/5, ... / ...|...] ;
Path = [2/2, 1/4, 3/5, 5/4, 3/3, 4/1, 5/3, 4/5, ... / ...|...] ;
Path = [5/3, 4/1, 2/2, 1/4, 3/5, 5/4, 3/3, 4/5, ... / ...|...] ;
Path = [4/4, 5/2, 3/1, 1/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [2/2, 4/1, 5/3, 4/5, 2/4, 1/2, 3/1, 5/2, ... / ...|...] ;
Path = [3/1, 5/2, 4/4, 2/5, 1/3, 2/1, 3/3, 1/2, ... / ...|...] ;
Path = [4/4, 2/5, 1/3, 2/1, 3/3, 5/2, 3/1, 1/2, ... / ...|...] ;
Path = [1/3, 2/1, 3/3, 2/5, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
Path = [3/3, 2/1, 1/3, 2/5, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
Path = [2/4, 1/2, 3/1, 5/2, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [5/3, 4/5, 2/4, 1/2, 3/1, 5/2, 4/4, 2/5, ... / ...|...] ;
Path = [3/5, 1/4, 2/2, 4/1, 5/3, 4/5, 2/4, 1/2, ... / ...|...] ;
Path = [4/2, 2/1, 1/3, 2/5, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
Path = [1/3, 2/5, 4/4, 5/2, 3/1, 1/2, 2/4, 4/5, ... / ...|...] ;
Path = [3/1, 1/2, 2/4, 4/5, 5/3, 4/1, 2/2, 1/4, ... / ...|...] ;
Path = [2/4, 4/5, 5/3, 4/1, 2/2, 1/4, 3/3, 1/2, ... / ...|...] ;
Path = [3/3, 1/4, 2/2, 4/1, 5/3, 4/5, 2/4, 1/2, ... / ...|...] ;
Path = [2/2, 1/4, 3/3, 4/1, 5/3, 4/5, 2/4, 1/2, ... / ...|...] ;
Path = [5/3, 4/1, 2/2, 1/4, 3/3, 4/5, 2/4, 1/2, ... / ...|...] ;
Path = [4/4, 5/2, 3/1, 1/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [2/2, 4/1, 5/3, 4/5, 2/4, 1/2, 3/1, 5/2, ... / ...|...] ;
Path = [3/1, 5/2, 4/4, 2/5, 1/3, 2/1, 4/2, 5/4, ... / ...|...] ;
Path = [4/4, 2/5, 1/3, 2/1, 4/2, 5/4, 3/3, 5/2, ... / ...|...] ;
Path = [1/3, 2/1, 4/2, 5/4, 3/3, 2/5, 4/4, 5/2, ... / ...|...] ;
Path = [3/3, 5/4, 4/2, 2/1, 1/3, 2/5, 4/4, 5/2, ... / ...|...] ;
Path = [4/2, 5/4, 3/3, 2/1, 1/3, 2/5, 4/4, 5/2, ... / ...|...] ;
Path = [2/4, 1/2, 3/1, 5/2, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [5/3, 4/5, 2/4, 1/2, 3/1, 5/2, 4/4, 2/5, ... / ...|...] ;
Path = [4/4, 2/5, 1/3, 2/1, 4/2, 5/4, 3/5, 1/4, ... / ...|...] ;
Path = [1/3, 2/1, 4/2, 5/4, 3/5, 1/4, 2/2, 4/1, ... / ...|...] ;
```



```
Path = [3/3, 5/4, 4/2, 2/1, 1/3, 2/5, 4/4, 5/2, ... / ...|...] ;
Path = [4/2, 5/4, 3/3, 2/1, 1/3, 2/5, 4/4, 5/2, ... / ...|...] ;
Path = [2/4, 1/2, 3/1, 5/2, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [5/3, 4/5, 2/4, 1/2, 3/1, 5/2, 4/4, 2/5, ... / ...|...] ;
Path = [4/4, 2/5, 1/3, 2/1, 4/2, 5/4, 3/5, 1/4, ... / ...|...] ;
Path = [1/3, 2/1, 4/2, 5/4, 3/5, 1/4, 2/2, 4/1, ... / ...|...] ;
Path = [2/2, 4/1, 5/3, 4/5, 2/4, 1/2, 3/3, 1/4, ... / ...|...] ;
Path = [3/3, 1/2, 2/4, 4/5, 5/3, 4/1, 2/2, 1/4, ... / ...|...] ;
Path = [2/4, 1/2, 3/3, 4/5, 5/3, 4/1, 2/2, 1/4, ... / ...|...] ;
Path = [5/3, 4/5, 2/4, 1/2, 3/3, 4/1, 2/2, 1/4, ... / ...|...] ;
Path = [3/5, 1/4, 2/2, 4/1, 5/3, 4/5, 2/4, 1/2, ... / ...|...] ;
Path = [4/2, 5/4, 3/5, 1/4, 2/2, 4/1, 5/3, 4/5, ... / ...|...] ;
Path = [2/4, 4/5, 5/3, 4/1, 2/2, 1/4, 3/5, 5/4, ... / ...|...] ;
Path = [4/2, 2/1, 1/3, 2/5, 4/4, 5/2, 3/3, 5/4, ... / ...|...] ;
Path = [1/3, 2/5, 4/4, 5/2, 3/3, 2/1, 4/2, 5/4, ... / ...|...] ;
Path = [3/3, 5/2, 4/4, 2/5, 1/3, 2/1, 4/2, 5/4, ... / ...|...] ;
Path = [4/4, 5/2, 3/3, 2/5, 1/3, 2/1, 4/2, 5/4, ... / ...|...] ;
Path = [3/5, 5/4, 4/2, 2/1, 1/3, 2/5, 4/4, 5/2, ... / ...|...] ;
Path = [2/2, 1/4, 3/5, 5/4, 4/2, 2/1, 1/3, 2/5, ... / ...|...] ;
Path = [5/3, 4/1, 2/2, 1/4, 3/5, 5/4, 4/2, 2/1, ... / ...|...] ;
Path = [1/5, 2/3, 3/1, 1/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [1/5, 2/3, 4/2, 2/1, 3/3, 5/4, 3/5, 1/4, ... / ...|...] ;
Path = [1/5, 2/3, 3/5, 5/4, 4/2, 2/1, 3/3, 1/4, ... / ...|...] ;
Path = [1/5, 2/3, 4/4, 5/2, 3/1, 1/2, 2/4, 4/5, ... / ...|...] ;
Path = [1/5, 2/3, 3/1, 5/2, 4/4, 2/5, 3/3, 1/2, ... / ...|...] ;
Path = [1/5, 2/3, 4/4, 2/5, 3/3, 5/2, 3/1, 1/2, ... / ...|...] ;
Path = [1/5, 2/3, 3/5, 1/4, 2/2, 4/1, 5/3, 4/5, ... / ...|...] ;
Path = [1/5, 2/3, 4/2, 5/4, 3/5, 1/4, 2/2, 4/1, ... / ...|...] ;
Path = [1/5, 2/3, 3/1, 1/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [1/5, 2/3, 3/5, 5/4, 3/3, 1/4, 2/2, 4/1, ... / ...|...] ;
Path = [1/5, 2/3, 4/4, 5/2, 3/1, 1/2, 2/4, 4/5, ... / ...|...] ;
Path = [1/5, 2/3, 3/1, 5/2, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [1/5, 2/3, 4/4, 2/5, 1/3, 2/1, 3/3, 5/2, ... / ...|...] ;
Path = [1/5, 2/3, 3/5, 1/4, 2/2, 4/1, 5/3, 4/5, ... / ...|...] ;
Path = [1/5, 2/3, 3/1, 5/2, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [1/5, 2/3, 4/4, 2/5, 1/3, 2/1, 4/2, 5/4, ... / ...|...] ;
Path = [1/5, 2/3, 3/5, 1/4, 3/3, 5/4, 4/2, 2/1, ... / ...|...] ;
Path = [1/5, 2/3, 4/2, 5/4, 3/5, 1/4, 3/3, 2/1, ... / ...|...] ;
Path = [1/5, 2/3, 4/2, 2/1, 1/3, 2/5, 4/4, 5/2, ... / ...|...] ;
Path = [1/5, 2/3, 3/1, 1/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [1/5, 2/3, 4/4, 5/2, 3/1, 1/2, 2/4, 4/5, ... / ...|...] ;
Path = [1/5, 2/3, 3/5, 5/4, 4/2, 2/1, 1/3, 2/5, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 4/2, 2/1, 1/3, 2/5, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 3/1, 1/2, 2/4, 4/5, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 3/1, 5/2, 4/4, 2/5, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 4/2, 5/4, 3/3, 2/1, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 3/5, 1/4, 2/2, 4/1, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 4/2, 5/4, 3/5, 1/4, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 4/2, 2/1, 1/3, 2/5, ... / ...|...] ;
Path = [5/5, 3/4, 1/5, 2/3, 4/4, 5/2, 3/3, 2/5, ... / ...|...] ;
```



```
Path = [1/3, 2/1, 4/2, 5/4, 3/5, 1/4, 2/2, 4/1, ... / ...|...] ;
Path = [2/2, 4/1, 5/3, 4/5, 3/3, 1/4, 3/5, 5/4, ... / ...|...] ;
Path = [3/3, 4/5, 5/3, 4/1, 2/2, 1/4, 3/5, 5/4, ... / ...|...] ;
Path = [5/3, 4/5, 3/3, 4/1, 2/2, 1/4, 3/5, 5/4, ... / ...|...] ;
Path = [3/5, 1/4, 2/2, 4/1, 5/3, 4/5, 3/3, 5/4, ... / ...|...] ;
Path = [4/2, 5/4, 3/5, 1/4, 2/2, 4/1, 5/3, 4/5, ... / ...|...] ;
Path = [5/1, 3/2, 1/3, 2/5, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
Path = [5/1, 3/2, 2/4, 4/5, 5/3, 4/1, 2/2, 1/4, ... / ...|...] ;
Path = [5/1, 3/2, 5/3, 4/1, 2/2, 1/4, 3/3, 4/5, ... / ...|...] ;
Path = [5/1, 3/2, 4/4, 5/2, 3/1, 1/2, 2/4, 4/5, ... / ...|...] ;
Path = [5/1, 3/2, 4/4, 2/5, 1/3, 2/1, 4/2, 5/4, ... / ...|...] ;
Path = [5/1, 3/2, 1/3, 2/1, 4/2, 5/4, 3/3, 2/5, ... / ...|...] ;
Path = [5/1, 3/2, 2/4, 1/2, 3/1, 5/2, 4/4, 2/5, ... / ...|...] ;
Path = [5/1, 3/2, 5/3, 4/5, 2/4, 1/2, 3/1, 5/2, ... / ...|...] ;
Path = [5/1, 3/2, 4/4, 2/5, 1/3, 2/1, 4/2, 5/4, ... / ...|...] ;
Path = [5/1, 3/2, 1/3, 2/1, 4/2, 5/4, 3/5, 1/4, ... / ...|...] ;
Path = [5/1, 3/2, 2/4, 1/2, 3/3, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [5/1, 3/2, 5/3, 4/5, 2/4, 1/2, 3/3, 4/1, ... / ...|...] ;
Path = [5/1, 3/2, 2/4, 4/5, 5/3, 4/1, 2/2, 1/4, ... / ...|...] ;
Path = [5/1, 3/2, 1/3, 2/5, 4/4, 5/2, 3/3, 2/1, ... / ...|...] ;
Path = [5/1, 3/2, 4/4, 5/2, 3/3, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [5/1, 3/2, 5/3, 4/1, 2/2, 1/4, 3/5, 5/4, ... / ...|...] ;
;;Path = [5/5, 4/3, 5/1, 3/2, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 1/3, 2/1, 4/2, 5/4, ... / ...|...] ;
;Path = [5/5, 4/3, 5/1, 3/2, 2/4, 1/2, 3/1, 5/2, ... / ...|...] ;
;Path = [5/5, 4/3, 5/1, 3/2, 1/3, 2/5, 4/4, 5/2, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 2/4, 4/5, 3/3, 1/2, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
;;Path = [5/5, 4/3, 5/1, 3/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
;Path = [5/5, 4/3, 5/1, 3/2, 5/3, 4/1, 2/2, 1/4, ... / ...|...] ;
;Path = [5/5, 4/3, 5/1, 3/2, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 4/4, 2/5, 3/3, 5/2, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 2/4, 1/2, 3/1, 5/2, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 5/3, 4/5, 2/4, 1/2, ... / ...|...] ;
;Path = [5/5, 4/3, 5/1, 3/2, 1/3, 2/5, 4/4, 5/2, ... / ...|...] ;
;;Path = [5/5, 4/3, 5/1, 3/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 5/3, 4/1, 2/2, 1/4, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
;Path = [5/5, 4/3, 5/1, 3/2, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 1/3, 2/1, 3/3, 2/5, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 2/4, 1/2, 3/1, 5/2, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 5/3, 4/5, 2/4, 1/2, ... / ...|...] ;
;;;;Path = [5/5, 4/3, 5/1, 3/2, 4/4, 2/5, 1/3, 2/1, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 1/3, 2/1, 4/2, 5/4, ... / ...|...] ;
;Path = [5/5, 4/3, 5/1, 3/2, 2/4, 1/2, 3/1, 5/2, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 5/3, 4/5, 2/4, 1/2, ... / ...|...] ;
;Path = [5/5, 4/3, 5/1, 3/2, 1/3, 2/5, 4/4, 5/2, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 2/4, 4/5, 5/3, 4/1, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 5/3, 4/1, 3/3, 4/5, ... / ...|...] ;
Path = [5/5, 4/3, 5/1, 3/2, 4/4, 5/2, 3/1, 1/2, ... / ...|...] ;
;false.

?- 
```



```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/CI LAB/LAB 3$ prolog q5.pl
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).

?- knight_tour(2, Path).
false.

?- 
```

For Code , refer to the GitHub:

https://github.com/arnavjain2710/Computational-Intelligence-Lab-CS354N/tree/main/LAB%203