Name: Dhruval Shah

ID: 200361439

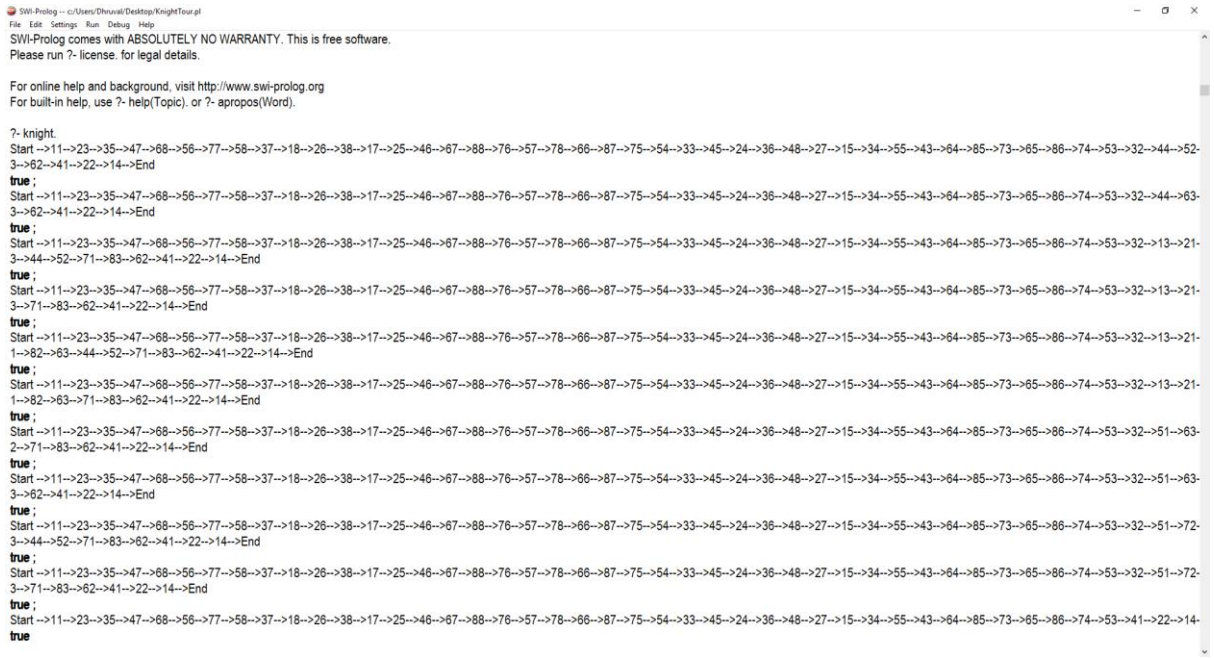
HW 9

Write PROLOG code to solve the full 8 x 8 knights' tour problem. Use the production system architecture proposed in this chapter and Chapter 6. Instead of square numbers of 1 to 64, use the pair of row and column numbers to indicate each square. Create the code so that you can get the program to solve a number of different tasks, such as getting from any one square to another.

a. Execute this code and draw a graph of the search space.

b. Alter the rule ordering to produce alternative solution paths. Reverse the order of the moves.

A. Execute this code and draw a graph of the search space.

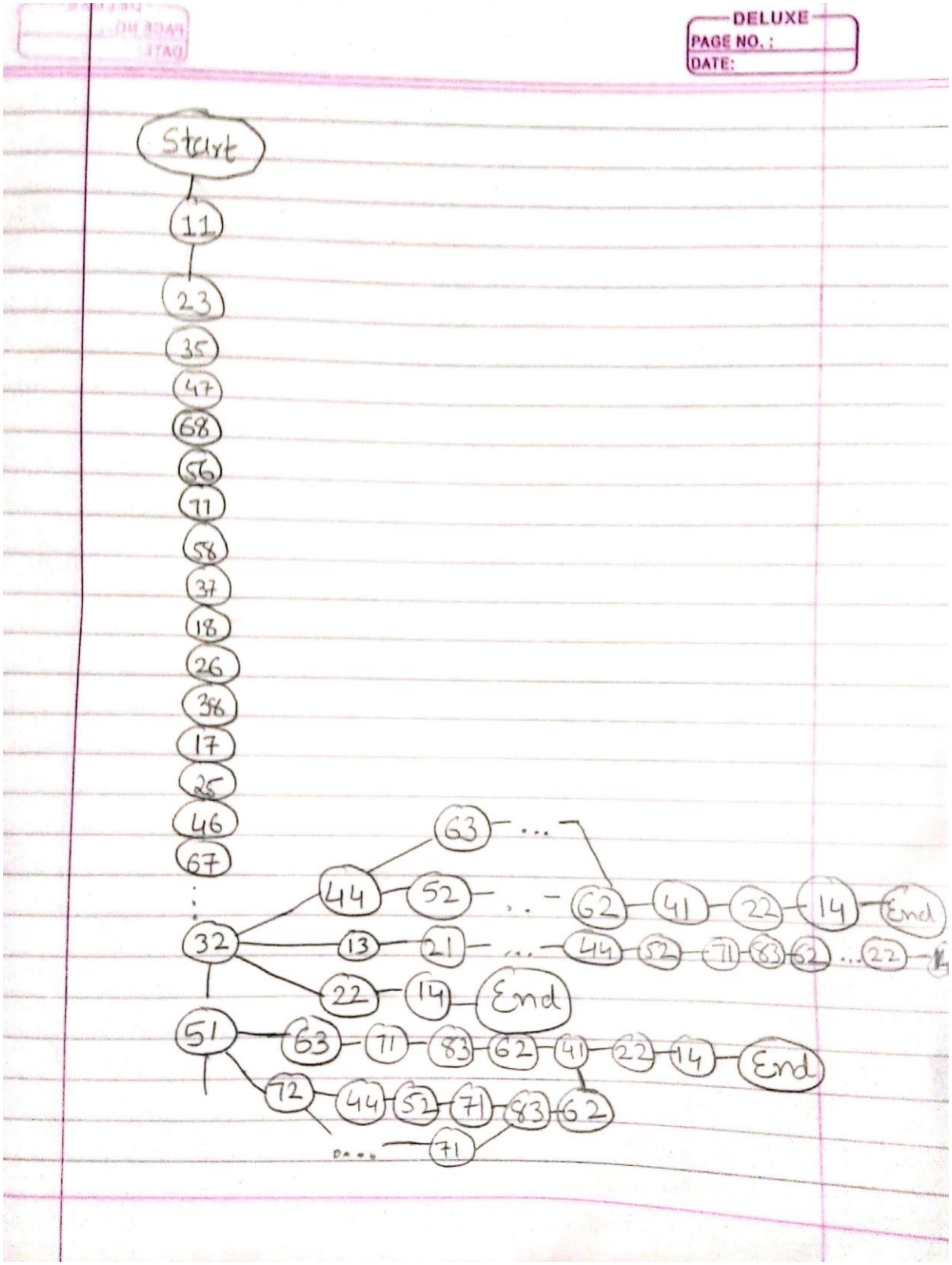


Program:

```
%Initial_state
initial_state([state(1,1)]).

%Goal_state
goal_state([state(1,4)]).

% defining base confition,
board(Coord) :-
      0 < Coord,
      Coord < 9.

% Here function show display all the states step by step.
show([]).
```

```
show([state(S,G)|Later]) :-
                write(S),
                write(G),
                write(-->),
                show(Later).

% divide input into two.
div(I,[I|_]).
div(I,[_|J]) :- div(I,J).

% Reverse the list
rev([],List,List).
rev([Head|Tail],L,List) :- rev(Tail,[Head|L],List).

% Check goal state is found or not
solve([state(1,4)|Before],[state(1,4)|Before]).

% Define possible places where knight can move from its current state
solve([state(S1,G1)|Before],RawAns) :-
      div([S,G],[[1,2],[2,1],[-2,1],[-1,2],[-2,-1],[-1,-2],[1,-2],[2,-1]]),
      S2 is S1 + S,
      board(S2),
      G2 is G1 + G,
      board(G2),
      %S2 <= 7, G2 <=7,
      %S2 >=0, G2 >=0,
      \+ div(state(S2,G2),Before),
      solve([state(S2,G2),state(S1,G1) |Before],RawAns).

knight :-
      initial_state(Start),
      goal_state(_),
      solve(Start,RawAns),
      rev(RawAns,[],FinalAns),
      write('Start -->'),
      show(FinalAns),
      write('End').
```

State Space:



B. Alter the rule ordering to produce alternative solution paths. Reverse the order of the moves.

Program after changing order of moves

```
%Initial_state
initial_state([state(1,1)]).

%Goal_state
goal_state([state(1,4)]).

% defining base confition,
board(Coord) :-
    0 < Coord,
    Coord < 9.

% Here function show display all the states step by step.
show([]).
show([state(S,G)|Later]) :-
            write(S),
            write(G),
            write(-->),
            show(Later).

% divide input into two.
div(I,[I|_]).
div(I,[_|J]) :- div(I,J).

% Reverse the list
rev([],List,List).
rev([Head|Tail],L,List) :- rev(Tail,[Head|L],List).

% Check goal state is found or not
solve([state(1,4)|Before],[state(1,4)|Before]).

% Define possible places where knight can move from its current state
solve([state(S1,G1)|Before],RawAns) :-
    div([S,G],[[2,-1],[1,-2],[-1,-2],[-2,-1],[-1,2],[-2,1],[2,1],[1,2]]),
    S2 is S1 + S,
    board(S2),
    G2 is G1 + G,
    board(G2),
    %S2 <= 7, G2 <=7,
    %S2 >=0, G2 >=0,
    \+ div(state(S2,G2),Before),
    solve([state(S2,G2),state(S1,G1) |Before],RawAns).

knight :-
    initial_state(Start),
    goal_state(_),
    solve(Start,RawAns),
    rev(RawAns,[],FinalAns),
    write('Start -->'),
    show(FinalAns),
    write('End').
```

SWI-Prolog -- c:/Users/Dhruval/Desktop/KnightTour.pl

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- knight.
Start -->11-->32-->51-->43-->62-->81-->73-->61-->53-->72-->64-->83-->71-->63-->82-->74-->66-->85-->77-->65-->84-->76-->55-->34-->42-->21-->13-->25-->44-->52-->31-->23-->15-->36-->24-->12-->33-->41-->22-->14-->End
true ;
;Start -->11-->32-->51-->43-->62-->81-->73-->61-->53-->72-->64-->83-->71-->63-->82-->74-->66-->85-->77-->65-->84-->76-->55-->34-->42-->21-->13-->25-->44-->52-->31-->23-->15-->36-->24-->12-->33-->14-->End
true
Unknown action: (h for help)
Action? ;
Start -->11-->32-->51-->43-->62-->81-->73-->61-->53-->72-->64-->83-->71-->63-->82-->74-->66-->85-->77-->65-->84-->76-->55-->34-->42-->21-->13-->25-->44-->52-->31-->23-->15-->36-->24-->12-->33-->54-->46-->38-->57-->45-->37-->56-
7-->48-->27-->35-->14-->End
true ;
Start -->11-->32-->51-->43-->62-->81-->73-->61-->53-->72-->64-->83-->71-->63-->82-->74-->66-->85-->77-->65-->84-->76-->55-->34-->42-->21-->13-->25-->44-->52-->31-->23-->15-->36-->24-->12-->33-->54-->46-->38-->57-->45-->37-->56-
7-->48-->27-->35-->16-->28-->47-->26-->14-->End
true ;
Start -->11-->32-->51-->43-->62-->81-->73-->61-->53-->72-->64-->83-->71-->63-->82-->74-->66-->85-->77-->65-->84-->76-->55-->34-->42-->21-->13-->25-->44-->52-->31-->23-->15-->36-->24-->12-->33-->54-->46-->38-->57-->45-->37-->56-
7-->48-->27-->35-->47-->26-->14-->End
true ;
Start -->11-->32-->51-->43-->62-->81-->73-->61-->53-->72-->64-->83-->71-->63-->82-->74-->66-->85-->77-->65-->84-->76-->55-->34-->42-->21-->13-->25-->44-->52-->31-->23-->15-->36-->24-->12-->33-->54-->46-->38-->57-->45-->37-->56-
7-->68-->47-->35-->14-->End
true ;
Start -->11-->32-->51-->43-->62-->81-->73-->61-->53-->72-->64-->83-->71-->63-->82-->74-->66-->85-->77-->65-->84-->76-->55-->34-->42-->21-->13-->25-->44-->52-->31-->23-->15-->36-->24-->12-->33-->54-->46-->38-->57-->45-->37-->56-
7-->68-->47-->26-->14-->End
true ;
Start -->11-->32-->51-->43-->62-->81-->73-->61-->53-->72-->64-->83-->71-->63-->82-->74-->66-->85-->77-->65-->84-->76-->55-->34-->42-->21-->13-->25-->44-->52-->31-->23-->15-->36-->24-->12-->33-->54-->46-->38-->57-->45-->37-->56-
7-->68-->47-->28-->16-->35-->14-->End
true ;
Start -->11-->32-->51-->43-->62-->81-->73-->61-->53-->72-->64-->83-->71-->63-->82-->74-->66-->85-->77-->65-->84-->76-->55-->34-->42-->21-->13-->25-->44-->52-->31-->23-->15-->36-->24-->12-->33-->54-->46-->38-->57-->45-->37-->56-
4-->End
true ;
Start -->11-->32-->51-->43-->62-->81-->73-->61-->53-->72-->64-->83-->71-->63-->82-->74-->66-->85-->77-->65-->84-->76-->55-->34-->42-->21-->13-->25-->44-->52-->31-->23-->15-->36-->24-->12-->33-->54-->46-->38-->57-->45-->37-->56-
7-->48-->67-->75-->87-->68-->47-->26-->14-->End
true .

?-
```