Batch: A1 **Experiment Number:** 6

Roll Number: 16010422012 **Name:** Ayaan Bhoje

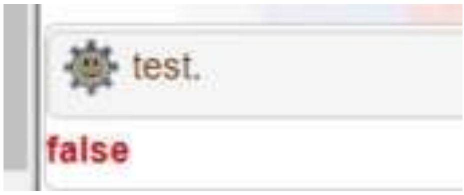
Aim of the Experiment: Write a program for implementation of Prolog program on 8-Puzzle

Program/ Steps:

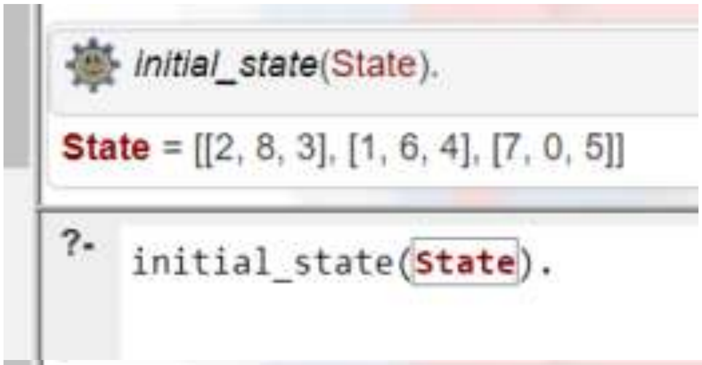
```
% Initial state of the puzzle
initial_state([
   [2, 8, 3],
   [1, 6, 4],
   [7, 0, 5]
]).

% Final state of the puzzle
final_state([
   [1, 2, 3],
   [8, 0, 4],
   [7, 6, 5]
]).

% Move puzzle piece from one position to another
move(State, NextState) :-
   select(Piece, State, Row),                  % Select a row
   select(0, Row, EmptyRow),                   % Find the empty cell (0)
   select(NewPiece, NextRow, EmptyRow),        % Select a new row
   replace(0, NewPiece, Row, NewRow),          % Replace the empty cell with the new piece
   replace(Piece, 0, NextRow, EmptyRow),       % Replace the new piece with the empty cell
   append([NewRow], NextRow, NextState).       % Append the new row to get the next
state

% Replace element in a list
replace(X, Y, [X|T], [Y|T]).
replace(X, Y, [H|T], [H|Z]) :-
   replace(X, Y, T, Z).

% Depth-first search
dfs(State, _, Path, Path) :-
   final_state(State).
dfs(State, Visited, Path, FinalPath) :-
   move(State, NextState),
   \+ member(NextState, Visited),              % Ensure we don't visit the same state again
```

```
    dfs(NextState, [NextState|Visited], [NextState|Path], FinalPath).

% Solve predicate
solve(Path) :-
    initial_state(InitialState),
    dfs(InitialState, [InitialState], [InitialState], Path).

% Test the program
test :-
    solve(Path),
    reverse(Path, Solution),
    write('Solution Path:'), nl,
    print_path(Solution).

% Print the solution path
print_path([]).
print_path([State|Rest]) :-
    print_board(State),
    nl,
    print_path(Rest).

% Print the board
print_board([]).
print_board([Row|Rest]) :-
    write(Row), nl,
    print_board(Rest).
```

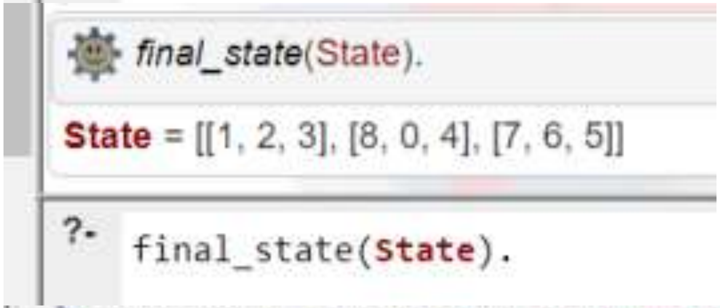
Output/Result:



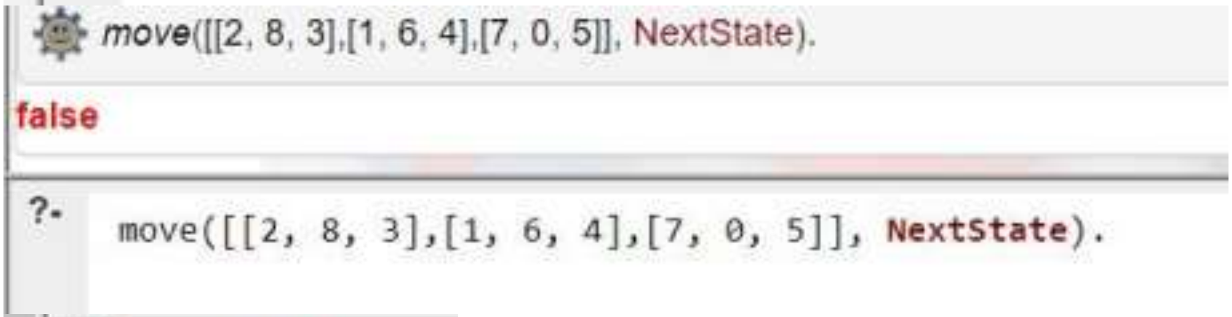
```
test.
false
```



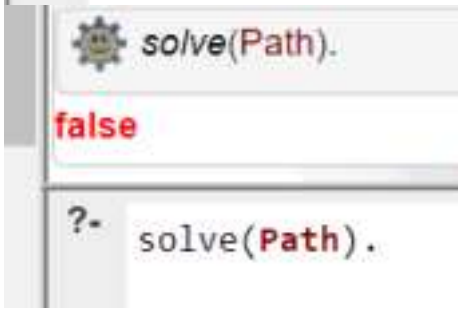
```
initial_state(State).

State = [[2, 8, 3], [1, 6, 4], [7, 0, 5]]

?- initial_state(State).
```



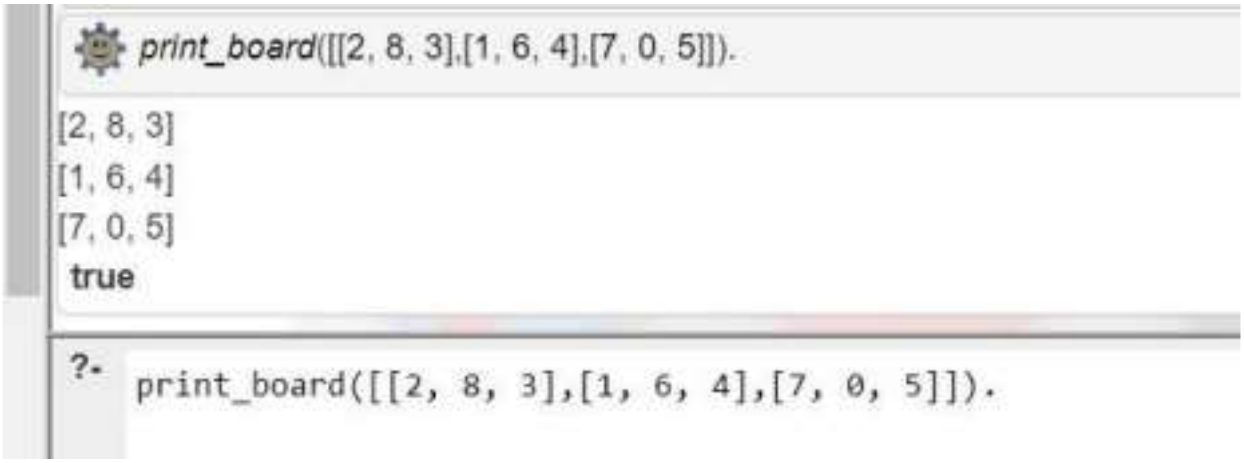
```
final_state(State).

State = [[1, 2, 3], [8, 0, 4], [7, 6, 5]]

?- final_state(State).
```



```
move([[2, 8, 3],[1, 6, 4],[7, 0, 5]], NextState).

false

?- move([[2, 8, 3],[1, 6, 4],[7, 0, 5]], NextState).
```



```
solve(Path).

false

?- solve(Path).
```



```
print_board([[2, 8, 3],[1, 6, 4],[7, 0, 5]]).
[2, 8, 3]
[1, 6, 4]
[7, 0, 5]
true

?- print_board([[2, 8, 3],[1, 6, 4],[7, 0, 5]]).
```

Outcomes:
CO-3: Ability to formally state the problem and develop the appropriate proof for a given logical deduction problem.

Conclusion (based on the Results and outcomes achieved):
Through this experiment, we learnt how to use prolog. We implemented the solution to solve the 8-puzzle problem using prolog.

References:

- Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Second Edition, Pearson Publication
- Luger, George F. Artificial Intelligence : Structures and strategies for complex problem solving , 2009 ,6th Edition, Pearson Education
- Ivan Bratko, Prolog Programming for AI, 2011, 4th Edition, Pearson publication