Ex.no : 05 Date:	Write a program to solve 8-puzzle problem

Aim:

To write a program to solve 8 – puzzle problems using prolog.

Algorithm

Step 1: Start the program and represent the board position as 8*8 vectors, i.e., [1,2,3,4,5,6,7.8]. Store the set of queens in the list 'Queens'.

Step 2: Calculate the permutation of the above eight numbers stored in set P.

Step 3: Let the position where the first queen to be placed be(1,Y), for second be (2,Y1),and so on and store the position in S.

Step 4: Check for the safety of the queens through the predicate, 'noattack()'.

Step 5: Calculate Y1-y and Y-Y1. If both are not equal to Xdist, which is the X- distance between the first queen and others, then goto step 6 else goto step 7.

Step 6: Increment Xdist by 1.

Step 7: Repeat above for the rest of the queens, until the end of the list is reached.

Step 8: Print S as answer.

Step 9: Stop the program.

Program:

```
test(Plan):-
write('Initial state:'),nl,
Init= [at(empty,1), at(tile8,2), at(tile1,3), at(tile5,4), at(tile3,5), at(tile2,6), at(tile7,7), at(tile4,8), at(tile6,9)],
write_sol(Init),
Goal= [at(tile1,1), at(tile2,2), at(tile3,3), at(tile4,4), at(tile5,5), at(tile6,6), at(tile7,7), at(tile8,8), at(empty,9)],
nl,write('Goal state:'),nl,
write(Goal),nl,nl,
solve(Init,Goal,Plan).
solve(State, Goal, Plan):-
solve(State, Goal, [], Plan).
is_movable(X1,Y1) :- (1 is X1 - Y1) ; (-1 is X1 - Y1) ; (3 is X1 - Y1) ; (-3 is X1 - Y1).
solve(State, Goal, Plan, Plan):-
is_subset(Goal, State), nl,
write_sol(Plan).
solve(State, Goal, Sofar, Plan):-
```

```
act(Action, Preconditions, Delete, Add),
is_subset(Preconditions, State),
\+ member(Action, Sofar), delete_list(Delete,
State, Remainder), append(Add, Remainder,
NewState), solve(NewState, Goal,
[Action|Sofar], Plan). act(move(X,Y,Z),
[at(X,Y), at(empty,Z), is_movable(Y,Z)],
[at(X,Y), at(empty,Z)],
[at(X,Z), at(empty,Y)]).
is_subset([H|T], Set):-
member(H, Set),
is_subset(T, Set).
is_subset([], _).
delete_list([H|T], Curstate, Newstate):-
remove(H, Curstate, Remainder),
delete_list(T, Remainder, Newstate).
delete_list([], Curstate, Curstate).
remove(X, [X|T], T).
remove(X, [H|T], [H|R]):-
remove(X, T, R).
write_sol([]).
write_sol([H|T]):-
write_sol(T), write(H),
nl.
append([H|T], L1, [H|L2]):-
append(T, L1, L2).
append([], L, L).
member(X, [X|_]).
member(X, [_|T]):-
member(X, T).
```

Initial State

	8	1
5	3	2
7	4	6

Goal State

1	2	3
4	5	6
7	8	

Output Screenshot:

```
?-
% c:/users/administrator.dsp-43/documents/prolog/puzzle8 compiled 0.00 sec, 0 clauses
?-
|    test(Plan).
Initial state:
at(tile6,9)
at(tile4,8)
at(tile7,7)
at(tile2,6)
at(tile3,5)
at(tile5,4)
at(tile1,3)
at(tile8,2)
at(empty,1)

Goal state:
[at(tile1,1),at(tile2,2),at(tile3,3),at(tile4,4),at(tile5,5),at(tile6,6),at(tile7,7),at(tile8,8),at(empty,9)]

false.
```

Result:

Thus the program for the 8 puzzle problem using prolog is executed and the output is obtained successfully.