exp6. 8 puzzle problem

```
test(Plan):-

write('Initial state:'),nl,

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

write_sol(Init),

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

nl,write('Goal state:'),nl,

write(Goal),nl,nl,

solve(Init,Goal,Plan).

solve(State, Goal, Plan):-

solve(State, Goal, [], Plan).

% Determines whether Current and Destination tiles are a valid move.

is_movable(X1,Y1) :- (1 is X1 - Y1) ; (-1 is X1 - Y1) ; (3 is X1 - Y1) ; (-3 is X1 - Y1).

/* This predicate produces the plan. Once the Goal list is a subset of the current State the plan is complete and it is written to the screen using write_sol */

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).

/* The problem has three operators.

1st arg = name

2nd arg = preconditions

3rd arg = delete list

4th arg = add list. */
```

```
% Tile can move to new position only if the destination tile is empty & Manhattan distance = 1
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)]).
% Utility predicates.
% Check is first list is a subset of the second
is_subset([H|T], Set):-
member(H, Set),
is_subset(T, Set).
is_subset([], _).
% Remove all elements of 1st list from second to create third.
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).
```