Ex.no : 10 Date:	Write a program to solve missionaries and cannibal problem

Aim:

To write a prolog program to solve Missionaries and Cannibal Problem.

Procedure:

The Missionaries and Cannibals problem is a classic problem in Artificial Intelligence that involves three missionaries and three cannibals on one side of a river, along with a boat that can carry at most two people at a time. The goal is to move all the people to the other side of the river without ever leaving more missionaries than cannibals on either side, or else the cannibals will eat the missionaries.

Program:

```
% Represent a state as [CL,ML,B,CR,MR]
start([3,3,left,0,0]).
goal([0,0,right,3,3]).
legal(CL,ML,CR,MR) :-
% is this state a legal one? ML>=0,
CL>=0, MR>=0, CR>=0,
(ML>=CL ; ML=0),
(MR>=CR ; MR=0).
% Possible moves:
move([CL,ML,left,CR,MR],[CL,ML2,right,CR,MR2]):-
% Two missionaries cross left to right.
MR2 is MR+2,
ML2 is ML-2,
legal(CL,ML2,CR,MR2).
move([CL,ML,left,CR,MR],[CL2,ML,right,CR2,MR]):-
% Two cannibals cross left to right.
CR2 is CR+2,
CL2 is CL-2,
legal(CL2,ML,CR2,MR).
move([CL,ML,left,CR,MR],[CL2,ML2,right,CR2,MR2]):-
% One missionary and one cannibal cross left to right.
CR2 is CR+1,
CL2 is CL-1,
```

```
MR2 is MR+1,
ML2 is ML-1,
legal(CL2,ML2
,CR2,MR2).
move([CL,ML,left,CR,MR],[CL,ML2,right,CR,MR2]):-
% One missionary crosses left to right.
MR2 is MR+1,
ML2 is ML-1,
legal(CL,ML2,CR,MR2).
move([CL,ML,left,CR,MR],[CL2,ML,right,CR2,MR]):-
% One cannibal crosses left to right.
CR2 is CR+1,
CL2 is CL-1,
legal(CL2,ML,CR2,MR).
move([CL,ML,right,CR,MR],[CL,ML2,left,CR,MR2]):-
% Two missionaries cross right to left.
MR2 is MR-2,
ML2 is ML+2,
legal(CL,ML2,CR,MR2).
move([CL,ML,right,CR,MR],[CL2,ML,left,CR2,MR]):-
% Two cannibals cross right to left.
CR2 is CR-2,
CL2 is CL+2,
legal(CL2,ML,CR2,MR).
move([CL,ML,right,CR,MR],[CL2,ML2,left,CR2,MR2]):-
% One missionary and one cannibal cross right to left.
CR2 is CR-1,
CL2 is CL+1,
MR2 is MR-1,
ML2 is ML+1,
legal(CL2,ML2,CR2,MR2).
move([CL,ML,right,CR,MR],[CL,ML2,left,CR,MR2]):-
% One missionary crosses right to left.
MR2 is MR-1,
ML2 is ML+1,
legal(CL,ML2,CR,MR2).
move([CL,ML,right,CR,MR],[CL2,ML,left,CR2,MR]):-
% One cannibal crosses right to left.
CR2 is CR-1,
CL2 is CL+1,
```

```
legal(CL2,ML,CR2,MR).
% Recursive call to solve the problem
path([CL1,ML1,B1,CR1,MR1],[CL2,ML2,B2,CR2,MR2],Explored,MovesList) :-
move([CL1,ML1,B1,CR1,MR1],[CL3,ML3,B3,CR3,MR3]),
not(member([CL3,ML3,B3,CR3,MR3],Explored)),
path([CL3,ML3,B3,CR3,MR3],[CL2,ML2,B2,CR2,MR2],[[CL3,ML3,B3,CR3,MR3]|Explored],
[ [[CL3,ML3,B3,CR3,MR3],[CL1,ML1,B1,CR1,MR1]] | MovesList ]).
% Solution found
path([CL,ML,B,CR,MR],[CL,ML,B,CR,MR],_,MovesList):-
output(MovesList).
% Printing
output([]) :- nl.
output([[A,B]|MovesList]) :-
output(MovesList),
write(B), write(' -> '), write(A), nl.
% Find the solution for the missionaries and cannibals problem find:-
path([3,3,left,0,0],[0,0,right,3,3],[[3,3,left,0,0]],_).
```

Output Screenshot:

```
% c:/users/administrator.dell-3/documents/prolog/exp9man compiled 0.00 sec, 0 clauses
?- find.

[3,3,left,0,0] -> [1,3,right,2,0]
[1,3,right,2,0] -> [2,3,left,1,0]
[2,3,left,1,0] -> [0,3,right,3,0]
[0,3,right,3,0] -> [1,3,left,2,0]
[1,3,left,2,0] -> [1,1,right,2,2]
[1,1,right,2,2] -> [2,2,left,1,1]
[2,2,left,1,1] -> [2,0,right,1,3]
[2,0,right,1,3] -> [3,0,left,0,3]
[3,0,left,0,3] -> [1,0,right,2,3]
[1,0,right,2,3] -> [1,1,left,2,2]
[1,1,left,2,2] -> [0,0,right,3,3]
true .
```

Result:

Thus the program to solve missionaries and cannibal problem using prolog is executed and the output is obtained successfully.