```
:- set_prolog_flag(toplevel_print_options, [quoted(true),numbervars(true),portrayed(
true),max_depth(20)]).

start(state(3:3,boat_left, 0:0)).
goal(state(0:0,boat_right, 3:3)).

%Checks if element is not a member of a given list
nonm( _, [] ).
nonm( E, [H|T] ) :-
        dif( E, H ), nonm( E, T ).

%Depth-First search start clause.
df_search(Path) :-
        start(S0),
        df_search([S0],Path).

%Depth-First search goal clause
df_search([S|Visited],[S|Visited]) :-
        goal(S).

%Depth-First search general clause
df_search([S1|Visited], Path) :-
        action(S1,S2),
        %Make sure we doesn't visit a already visited state
        nonm(S2,[S1|Visited]),
        df_search([S2,S1|Visited], Path).

%Breath-First search start clause
bf_search(Path) :-
        start(S0),
        bf_search([[S0]],Path).
%Breath-First search goal clause
bf_search([[S|Path]|_], [S|Path]) :-
        goal(S).
%Breath-First search general clause
bf_search([[S1|Path]|Partials], FinalPath) :-
        %nonm added to prevent looping
        findall(S2,(action(S1,S2),nonm(S2,[S1|Path])), NewStates),
        expand([S1|Path], NewStates, NewPaths),
        append(Partials, NewPaths, NewPartials),
        bf_search(NewPartials,FinalPath).
%Generate all new paths from a given node
expand(L1,L2,L3) :-
        findall([X|L1], member(X,L2), L3).
%Make sure missionaries are same or outnumber the canibals
allowed_bank(M:C) :-
        M >= 0,
        C >= 0,
        M >= C.
%Extra cause to allow for banks with only canibals
allowed_bank(0:C) :-
        C > 0.
%Helper for allowed_banks
allowed_state(L,R) :-
        allowed_bank(L),
        allowed_bank(R).


% action - move one M from left to right
action(state(LM1:LC1, boat_left, RM1:RC1), state(LM2:LC1,boat_right,RM2:RC1)) :-
        LM2 is LM1 - 1,
        RM2 is RM1 + 1,
        allowed_state(LM2:LC1, RM2:RC1).
% action - move one C from left to right
action(state(LM1:LC1,boat_left, RM1:RC1), state(LM1:LC2, boat_right,RM1:RC2)) :-
        LC2 is LC1 - 1,
        RC2 is RC1 + 1,
        allowed_state(LM1:LC2, RM1:RC2).
% action - move one M from right to left
action(state(LM1:LC1,boat_right, RM1:RC1), state(LM2:LC1,boat_left,RM2:RC1)) :-
        RM2 is RM1 - 1,
        LM2 is LM1 + 1,
        allowed_state(LM2:LC1,RM2:RC1).
```

```
% action - move one C from right to left
action(state(LM1:LC1,boat_right, RM1:RC1), state(LM1:LC2,boat_left,RM1:RC2)) :-
        RC2 is RC1 - 1,
        LC2 is LC1 + 1,
        allowed_state(LM1:LC2,RM1:RC2).

% action - move one M from left to right
action(state(LM1:LC1,boat_left, RM1:RC1), state(LM2:LC1,boat_right,RM2:RC1)) :-
        LM2 is LM1 - 2,
        RM2 is RM1 + 2,
        allowed_state(LM2:LC1, RM2:RC1).
% action - move one C from left to right
action(state(LM1:LC1,boat_left,RM1:RC1), state(LM1:LC2,boat_right,RM1:RC2)) :-
        LC2 is LC1 - 2,
        RC2 is RC1 + 2,
        allowed_state(LM1:LC2, RM1:RC2).
% action - move one M from right to left
action(state(LM1:LC1,boat_right, RM1:RC1), state(LM2:LC1,boat_left,RM2:RC1)) :-
        RM2 is RM1 - 2,
        LM2 is LM1 + 2,
        allowed_state(LM2:LC1,RM2:RC1).
% action - move one C from right to left
action(state(LM1:LC1,boat_right,RM1:RC1), state(LM1:LC2,boat_left,RM1:RC2)) :-
        RC2 is RC1 - 2,
        LC2 is LC1 + 2,
        allowed_state(LM1:LC2,RM1:RC2).

% action - move one C from left to right
action(state(LM1:LC1,boat_right, RM1:RC1), state(LM2:LC2,boat_left,RM2:RC2)) :-
        RC2 is RC1 - 1,
        LC2 is LC1 + 1,
        RM2 is RM1 - 1,
        LM2 is LM1 + 1,
        allowed_state(LM2:LC2,RM2:RC2).
action(state(LM1:LC1,boat_left, RM1:RC1), state(LM2:LC2,boat_right,RM2:RC2)) :-
        RC2 is RC1 + 1,
        LC2 is LC1 - 1,
        RM2 is RM1 + 1,
        LM2 is LM1 - 1,
        allowed_state(LM2:LC2,RM2:RC2).


% In total there's 4 loop-free solutions, as can be viewed with the command findall(
X,df_search(X),P),length(P,L).

% Example  runs
%
% df_search(Path).
%Path = [state(0:0, boat_right, 3:3), state(1:1, boat_left, 2:2), state(0:1, boat_ri
ght, 3:2), state(0:3, boat_left, 3:0), state(0:2, boat_right, 3:1), state(2:2, boat_
left, 1:1), state(1:1, boat_right, 2:2), state(3:1, boat_left, 0:2), state(3:0, boat
_right, 0:3), state(3:2, boat_left, 0:1), state(3:1, boat_right, 0:2), state(3:3, bo
at_left, 0:0)] ;
%Path = [state(0:0, boat_right, 3:3), state(0:2, boat_left, 3:1), state(0:1, boat_ri
ght, 3:2), state(0:3, boat_left, 3:0), state(0:2, boat_right, 3:1), state(2:2, boat_
left, 1:1), state(1:1, boat_right, 2:2), state(3:1, boat_left, 0:2), state(3:0, boat
_right, 0:3), state(3:2, boat_left, 0:1), state(3:1, boat_right, 0:2), state(3:3, bo
at_left, 0:0)] ;
%Path = [state(0:0, boat_right, 3:3), state(1:1, boat_left, 2:2), state(0:1, boat_ri
ght, 3:2), state(0:3, boat_left, 3:0), state(0:2, boat_right, 3:1), state(2:2, boat_
left, 1:1), state(1:1, boat_right, 2:2), state(3:1, boat_left, 0:2), state(3:0, boat
_right, 0:3), state(3:2, boat_left, 0:1), state(2:2, boat_right, 1:1), state(3:3, bo
at_left, 0:0)] ;
%Path = [state(0:0, boat_right, 3:3), state(0:2, boat_left, 3:1), state(0:1, boat_ri
ght, 3:2), state(0:3, boat_left, 3:0), state(0:2, boat_right, 3:1), state(2:2, boat_
left, 1:1), state(1:1, boat_right, 2:2), state(3:1, boat_left, 0:2), state(3:0, boat
_right, 0:3), state(3:2, boat_left, 0:1), state(2:2, boat_right, 1:1), state(3:3, bo
at_left, 0:0)] ;
%no.
%
%bf_search(Path).
%Path = [state(0:0, boat_right, 3:3), state(1:1, boat_left, 2:2), state(0:1, boat_ri
ght, 3:2), state(0:3, boat_left, 3:0), state(0:2, boat_right, 3:1), state(2:2, boat_
left, 1:1), state(1:1, boat_right, 2:2), state(3:1, boat_left, 0:2), state(3:0, boat
_right, 0:3), state(3:2, boat_left, 0:1), state(3:1, boat_right, 0:2), state(3:3, bo
at_left, 0:0)] ;
%Path = [state(0:0, boat_right, 3:3), state(0:2, boat_left, 3:1), state(0:1, boat_ri
ght, 3:2), state(0:3, boat_left, 3:0), state(0:2, boat_right, 3:1), state(2:2, boat_
left, 1:1), state(1:1, boat_right, 2:2), state(3:1, boat_left, 0:2), state(3:0, boat
_right, 0:3), state(3:2, boat_left, 0:1), state(3:1, boat_right, 0:2), state(3:3, bo
at_left, 0:0)] ;
%Path = [state(0:0, boat_right, 3:3), state(1:1, boat_left, 2:2), state(0:1, boat_ri
ght, 3:2), state(0:3, boat_left, 3:0), state(0:2, boat_right, 3:1), state(2:2, boat_
left, 1:1), state(1:1, boat_right, 2:2), state(3:1, boat_left, 0:2), state(3:0, boat
_right, 0:3), state(3:2, boat_left, 0:1), state(2:2, boat_right, 1:1), state(3:3, bo
at_left, 0:0)] ;
%Path = [state(0:0, boat_right, 3:3), state(0:2, boat_left, 3:1), state(0:1, boat_ri
ght, 3:2), state(0:3, boat_left, 3:0), state(0:2, boat_right, 3:1), state(2:2, boat_
left, 1:1), state(1:1, boat_right, 2:2), state(3:1, boat_left, 0:2), state(3:0, boat
_right, 0:3), state(3:2, boat_left, 0:1), state(2:2, boat_right, 1:1), state(3:3, bo
at_left, 0:0)] ;
%no.
```