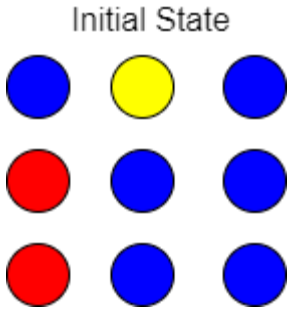
uninformed search

Design input & Design state representation:

We can represent the state as a 2D list that contains (N * M) elements, each element has a specific color and position which represents a node or a cell in the board.

For example, the initial state of the board in a problem may be as follows:

[[1, 1, blue], [1, 2, yellow], [1, 3, blue],
[2, 1, red], [2, 2, blue], [2, 3, blue],
[3, 1, red], [3, 2, blue], [3, 3, blue]]



Design moves:

```
% Movement rules within the grid.
move(State, Next, N, M):-
    left(State, Next, M);
    right(State, Next, M);
    up(State, Next, N);
    down(State, Next, N).

left([X,Y,_], [X1,Y1,_], M):-
    Y1 is Y-1, X1 is X, Y1 > 0, Y1 =< M.

right([X,Y,_], [X1,Y1,_], M):-
    Y1 is Y+1, X1 is X, Y1 > 0, Y1 =< M.

up([X,Y,_], [X1,Y1,_], N):-
    X1 is X-1, Y1 is Y, X1 > 0, X1 =< N.

down([X,Y,_], [X1,Y1,_], N):-
    X1 is X+1, Y1 is Y, X1 > 0, X1 =< N.
```

Design output:

The goal to search in the board using **DFS** and print at least one cycle including its color if more than one exists or no cycles exist if there are no cycles in the board.

As in the example, the goal is:

[2, 2, blue] => [2, 3, blue] => [3, 2, blue] => [3, 3, blue]