```
function [outPuzzle] = produce_random_puzzle(goalState, numStep,
 numPuzzle)

% Check if the input is a column vector othwerwise take transpose
if size(goalState, 1) == 1
    goalState = goalState';
end

outPuzzle = []; % Initialize the output

for iPuzzle = 1:numPuzzle
    currState = goalState; % Initialize current state as being the
 goal
    visitedStateMatrix = [];

    for iStep = 1:numStep
        visitedStateMatrix = [visitedStateMatrix currState];
        successorStates = successors(currState); % Generate successors
        numSucc = size(successorStates,2); % Number of the successors
        validSuccessors = []; % Store successors that can be visited
 for the next step

        for iSucc = 1:numSucc
            curSucc = successorStates(:, iSucc);

            if ~any(ismember(curSucc', visitedStateMatrix', 'rows'))
                validSuccessors = [validSuccessors curSucc];
            end
        end

        numValidSucc = size(validSuccessors, 2); % Number of valid
 successors
        k = floor(numValidSucc*rand) + 1; % Choose a random number
 between [1, numValidSucc]

        currState = validSuccessors(:, k); % Next state to be
 considered
    end

    outPuzzle = [outPuzzle currState];
end

end
```

Published with MATLAB® R2017b