```
% --------------------------------------------------------------------------
%
% A program that plays the n-puzzle where n > 2
%
% --------------------------------------------------------------------------
% ask the user for input for the kind of puzzle they want to play
while true
    n = input('Enter size n of the puzzle you want to play: ', 's');
    if (isnumeric(str2double(n)) && str2double(n) > 2)
        n = floor(str2double(n));
        break;
    else
        fprintf('Error: Invalid size, n must be an integer > 2.\n');
        continue;
    end
end

% --------------------------------------------------------------------------
% create a puzzle arranged correctly
correctPuzzle = puzzle(n, 1);
% create a randomized puzzle
while true
    randPuzzle = puzzle(n, 0);
    if ~isequal(randPuzzle, correctPuzzle)
        break;
    end
end

% display initial state
display(correctPuzzle, randPuzzle)

% get user to play by placing numbers in empty space
while true
    % get user input of number to move
    choice = input("Enter the digit you want to move or q to quit: ", "s");
    if ~ismember(randPuzzle, str2double(choice))
        if (choice == 'q')
            disp('Game terminated by User');
            break;
        end
        fprintf('Error: Invalid choice, try again.\n');
        continue;
    end
    choice = str2double(choice);

    % confirm that the number can be moved [next to empty space]
    [choiceR, choiceC] = find(randPuzzle == choice); % choice index
    [emptyR, emptyC] = find(randPuzzle == 0);      % empty Index
    if norm([choiceR - emptyR, choiceC - emptyC], 1) ~= 1
        fprintf('Error: %d is an Invalid move, try again.\n', choice);
        continue;
    end

    % move users choice to desired place [swap the two]
    randPuzzle(emptyR,emptyC) = choice;
    randPuzzle(choiceR, choiceC) = 0;

    % display current state of puzzle
    display(correctPuzzle, randPuzzle);

    % check if the game is complete
    if isequal(randPuzzle, correctPuzzle)
        disp('Congratulations, you have solved the puzzle');
        disp('----------------THE END --------------------')
```

```
        break;
    end
end
% ------------------------------------------------------------------------
% function to display game and correctly placed items
function display(correctPuzzle, randPuzzle)
    % display correctly placed tiles:
    correctlyPlacedTiles = sum(correctPuzzle(1:end, 1:end) == randPuzzle(1:end,
1:end), 'all');
    fprintf('Current Board: (#Correctly placed = %d)\n', correctlyPlacedTiles);
    % display new game position
    for i = 1:size(randPuzzle, 1)
        for j = 1:size(randPuzzle, 2)
            if (randPuzzle(i,j) == 0)
                fprintf('%s     ', '  ');
            else
                fprintf('%2d     ', randPuzzle(i,j));
            end
        end
        fprintf('\n\n');
    end
end

% ------------------------------------------------------------------------
% function to create a puzzle random or correct
function myPuzzle = puzzle(n, state)
    if (state == 0)                 % randomized puzzle
        myPuzzle = randperm(n^2)-1;
    else                            % normal puzzle
        myPuzzle = 1:(n^2) - 1;     % create an array from 1 to n^2-1
        myPuzzle(end+1) = 0;            % add an empty at the end
    end
    myPuzzle = reshape(myPuzzle, n, [])';   % create an n x n matrix
end
%------------------------------------------------------------------------
```