1 Carrayz 2D

1.1 Mirror Mirror on the Wall
The mirrored image should have the columns of RGB matrices flipped in right-left direction.

```matlab
lena = imread('lena.png');
lena(:,:,1) = lena(:,:,end:-1:1,:);
imshow(lena);
```

MATLAB has functions called `fliplr` and `flipud` to flip columns and rows.

1.2 Lena in Pieces
You can take out pieces of the matrices using a nested loop and computing the appropriate ranges. Since filename is a string, we can append a number to it after converting the number to a string using the function `num2str`.

```matlab
lena = imread('lena.png');
for i = 1:4
    for j = 1:4
        piece = lena(128*(i-1)+1:128*i,128*(j-1)+1:128*j,:);
        imwrite(piece, ['lena', num2str(4*(i-1)+j),'.png'],'png');
    end
end
```

If $m \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$, we can use the code in the previous session which were written for $m = 4$. What if we wanted $m = 3$ or $m = 5$ or $m = 21$? If you look up for help on function `imresize`, you will realize that MATLAB can help you enlarge or shrink the size of the image. The code will be similar, but first we will enlarge the image, and then shrink the pieces by rescaling.

```matlab
function lena_m2 (data, m)
    % assuming size(data) is 512 512 3
    n = lcm(512,m); % least common multiple
    s = n/m; % size of each piece
    enlarged = imresize(data, [n n]);
    for i = 1:m
        for j = 1:m
            piece = enlarged(s*(i-1)+1:s*i,s*(j-1)+1:s*j,:);
            piece = imresize(piece, 512/n); % scale back
            imwrite(piece, ['lena', num2str(m*(i-1)+j),'.png'],'png');
        end
    end
```

1.3 Scrambled Lena

The pieces are numbered starting from top left corner going to the right and then down. We will have to convert the piece number to a row and column number, and move data from the original image to the scrambled version.

```matlab
function J = scramble(I, S)
% We will use the scrambling matrix as a lookup to find
% the image data on the original image
J = I;
for i = 1:4
    for j = 1:4
        jj = rem(S(i,j)−1,4) + 1; % original block column location
        ii = floor((S(i,j)−jj)/4)+1; % original block row location
        J(128*(i−1)+1:128*i,128*(j−1)+1:128*j,:) = ...
            I(128*(ii−1)+1:128*ii,128*(jj−1)+1:128*jj,:);
    end
end
imshow(J)
```

We check the number of input arguments with the function `nargin`. If the scrambling matrix is not provided we can use functions `randperm` and `reshape` to produce one internally.

```matlab
function [J, S] = scramble(I, S)
if nargin < 2
    S = reshape(randperm(16),4,4);
end
J = I;
for i = 1:4
    for j = 1:4
        jj = rem(S(i,j)−1,4) + 1;
        ii = floor((S(i,j)−jj)/4)+1;
        J(128*(i−1)+1:128*i,128*(j−1)+1:128*j,:) = ...
            I(128*(ii−1)+1:128*ii,128*(jj−1)+1:128*jj,:);
    end
end
imshow(J)
```

1.4 Lena at Twilight

In the lab we used the figure window to find the pixel coordinates and color values on the images. We can find the center coordinates for the eyes and using an approximate radius we can set the green and blue layers inside these circles to zero.
lena = imread('lena.png');

xl = 266; yl = 266; % center of left eye
xr = 329; yr = 267; % center of right eye
r = 7; % radius of eyes

for i=250:280
    for j=255:275
        if (xl-j)^2 + (yl-i)^2 < r^2
            lena(i,j,2:3) = 0;
        end
    end
end
for j=320:340
    if (xr-j)^2 + (yr-i)^2 < r^2
        lena(i,j,2:3) = 0;
    end
end

imshow(lena);
imwrite(lena, 'lena_red_eyes.png', 'png');

1.5 15-Puzzle

The function names can’t start with numbers. That’s a reason why we named our function puzzle15.m.

First we should ask for the filename if it’s not provided. Then we will read the image and create the blank square.

```matlab
function puzzle15(fname)
    % Fifteen puzzle
    if nargin < 1
        fname = input('Enter filename: ','s');
    end
    I = imread(fname);
    I(end-127:end,end-127:end,:) = 255;
```

Then we will initialize a scrambling matrix and save it as the correct solution. We will modify the matrix $S$ as we move the pieces, and check against $S_0$ to see if the solution is reached.

```matlab
% Initialize the scrambling matrix
S = [1 2 3 4
     5 6 7 8
     9 10 11 12
     13 14 15 16];
S0 = S; % correct solution
We will generate many moves backwards and scramble $S$ matrix. Then we will show the scrambled image.

```matlab
% Generate N random moves
N = 1000;
moves = [];
r = 4;  % We will keep track of where the empty block is using r and c
c = 4;  % % empty block is using r and c
prev = 0;  % To remember previous move
while length(moves) < N
    switch ceil(4*rand)
    case 1
        if r < 4 && prev ~= 2
            moves = [moves, 'u'];
            $S(r,c)$ = $S(r+1,c)$;
            $S(r+1,c)$ = 16;
            r = r + 1;
        end
    case 2
        if r > 1 && prev ~= 1
            moves = [moves, 'd'];
            $S(r,c)$ = $S(r-1,c)$;
            $S(r-1,c)$ = 16;
            r = r - 1;
        end
    case 3
        if c < 4 && prev ~= 4
            moves = [moves, 'l'];
            $S(r,c)$ = $S(r,c+1)$;
            $S(r,c+1)$ = 16;
            c = c + 1;
        end
    case 4
        if c > 1 && prev ~= 3
            moves = [moves, 'r'];
            $S(r,c)$ = $S(r,c-1)$;
            $S(r,c-1)$ = 16;
            c = c - 1;
        end
    end
end

% Show the scrambled image
scramble(I,S);
```
Now we can start asking for user input. Based on the character user enters, we will update the scrambling matrix and show the new image, until the solution is reached. User can quit by pressing 'q'.

```matlab
% Let's play the game
[r, c] = find(S == 16);
while ˜isequal(S0, S)
    scramble(I,S);
    user = input('Enter a move (u,d,l,r) : ');
    switch user
    case 'u'
        if r < 4, S(r,c) = S(r+1,c); S(r+1,c) = 16; r = r + 1; end
    case 'd'
        if r > 1, S(r,c) = S(r-1,c); S(r-1,c) = 16; r = r - 1; end
    case 'l'
        if c < 4, S(r,c) = S(r,c+1); S(r,c+1) = 16; c = c + 1; end
    case 'r'
        if c > 1, S(r,c) = S(r,c-1); S(r,c-1) = 16; c = c - 1; end
    case 'q'
        return
    otherwise
        disp('Unknown move! Enter 'u','d','l','r' or 'q');
    end
end
When the solution is reached we will show red, green, and blue versions a couple of times. We used `pause` function to pause for some time given in seconds.

```matlab
imshow(I);
R = I; R(:,:,2:1:3)=0;
G = I; G(:,:,1:2:3)=0;
B = I; B(:,:,1:1:2)=0;
for j = [1:5 1:5]
    t = 0.5/j;
    imshow(R); pause(t);
    imshow(G); pause(t);
    imshow(B); pause(t);
end
imshow(I);
```