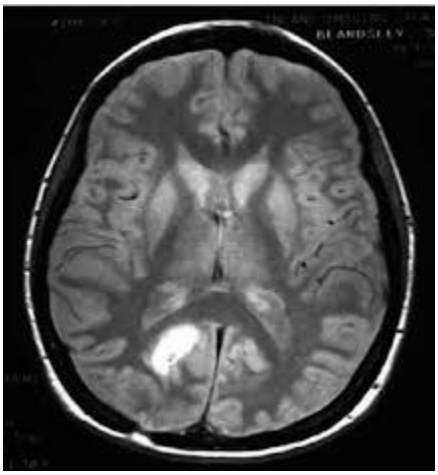
Homework_1 Image Processing

Group Partner: Abdullah Rihawi (2211011093)

Submitted by: Abdul Malik (2211011098)

Task 1: Image Reading and Matrix Representation

Image used:



Code covering parts 1,2 and 3:

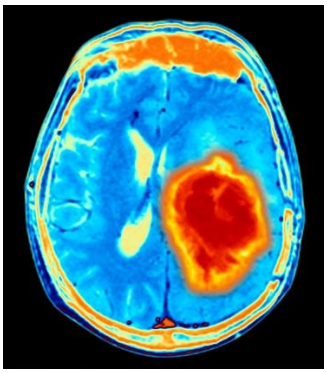
```
img = imread('image.jpg'); % Loads the image


imshow(img); % Displays the image


img_matrix = double(img); % Convert image to a matrix

disp(img_matrix); % Display numerical values
```

Explanation:

First we read the image through imread function of MATLAB, imshow function then displays the original image that we use to read before. Double function takes the complete image and converts it to the matrix form and finally disp function displays the matrix that has been constructed through double function before.

Task 2: Separating RGB Channels

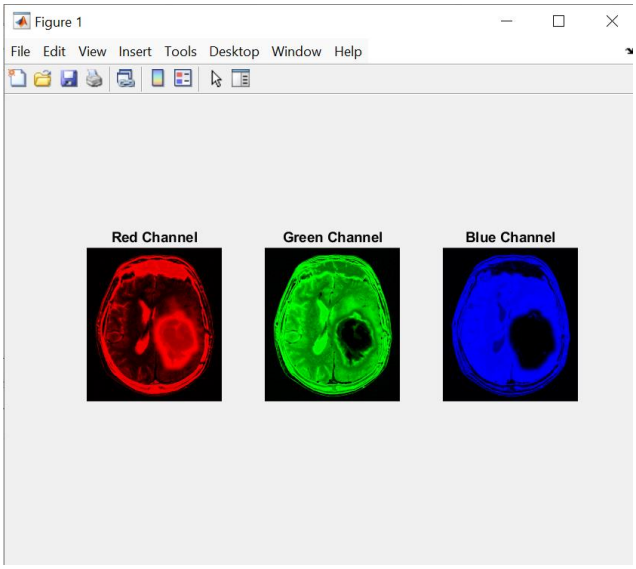
Image used:



Code:

```
img = imread(' 201007-Brain_tumour.jpg'); % Read the image


% Separate RGB channels

red_channel = img(:,:,1);

green_channel = img(:,:,2);

blue_channel = img(:,:,3);


% Display RGB channels correctly

figure;

subplot(1,3,1); imshow(cat(3, red_channel, zeros(size(img,1), size(img,2)), zeros(size(img,1), size(img,2)))); title('Red Channel');

subplot(1,3,2); imshow(cat(3, zeros(size(img,1), size(img,2)), green_channel, zeros(size(img,1), size(img,2)))); title('Green Channel');

subplot(1,3,3); imshow(cat(3, zeros(size(img,1), size(img,2)), zeros(size(img,1), size(img,2)), blue_channel)); title('Blue Channel');
```

Generated Image:



Explanation:

RGB images are 3D matrix but a 2D layer images and we can separate each color (red,green,blue) using the techniques in the above code where each color is being on and others are off in a function after separating and displaying each of them accurately. As we can see in our generated image, the representation of each color (Red, Green, Blue) being on respectively is shown when applied to our original image.

Task 3: Summary and Analysis

Write-up:

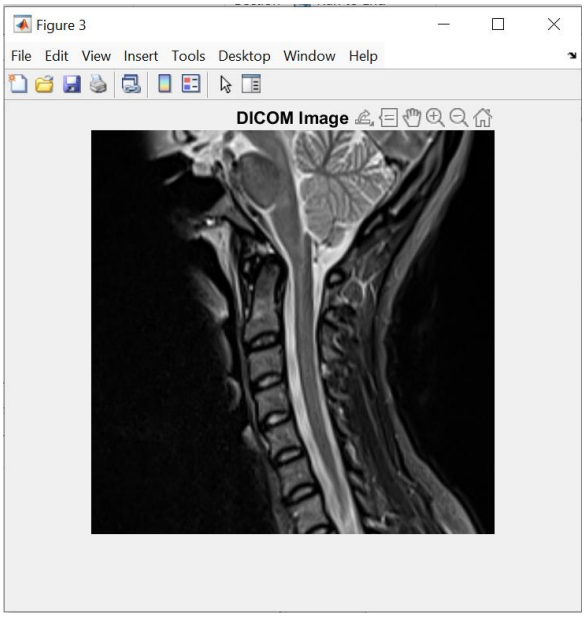
It is a tool used to represent matrices form of the images to help in doing mathematical operations on images. Separating RGB helps in understanding each color's density and thus easy to perform manipulative operations on the image to get the desired results.

Applications:

It used in so many medical diagnosing techniques including but not limited to MRI, CT Scane, PET etc. It is also used in computer vision which can be understand as object detection and facial recognition. Video processing also uses matrix representation.

Task 4: DICOM Image Reading, Display, and Metadata Extraction

Image used:



(After extraction using MATLAB)

Code:

```
% Read and display DICOM image
dicom_img = dicomread('image4.dcm');
info = dicominfo('image4.dcm');
figure;
imshow(dicom_img, []);
title('DICOM Image');

% Extract metadata
disp('Patient ID: '); disp(info.PatientID);
disp('Modality: '); disp(info.Modality);
```

Explanation:

The DICOM images have some hidden information that we call metadata which is relevant to the application. For example, in our case when the DICOM image was run using the MATLAB, we got the metadata as:

Patient ID:

2021.01.13-10:04:24-STD-1.3.12.2.1107.5.99.3

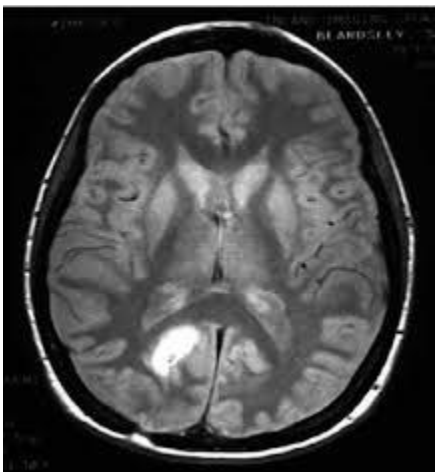
Modality:

MR

When we tried using another DICOM Image, it shows a different metadata.

Task 5: Basic Histogram Equalization

Image (same as we used in Task 1):



Code:

```
% Read a grayscale biomedical image
I = imread('image.jpg');

% Perform histogram equalization
I_eq = histeq(I);

% Display original and equalized images
figure;
subplot(1,2,1);
imshow(I);
```

```
title('Original Image');


subplot(1,2,2);
imshow(I_eq);
title('Histogram Equalized Image');
```

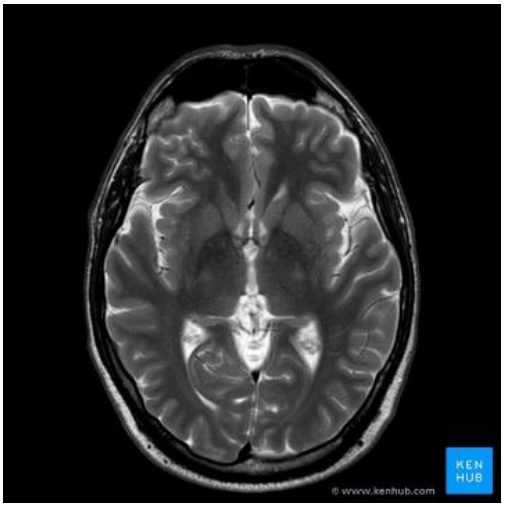
Histogram Equalized Image:

Explanation and Observations:

Histogram equalization redistributes the grayscale values so that the histogram of the resulting image is (approximately) uniform. This increases global contrast, making details in darker or lighter regions more visible. As a result, we can see that equalized image typically has enhanced contrast across the entire intensity range. Furthermore, previously indistinct details may become more pronounced.

Task 6: Brightness and Contrast Adjustment

Image used:

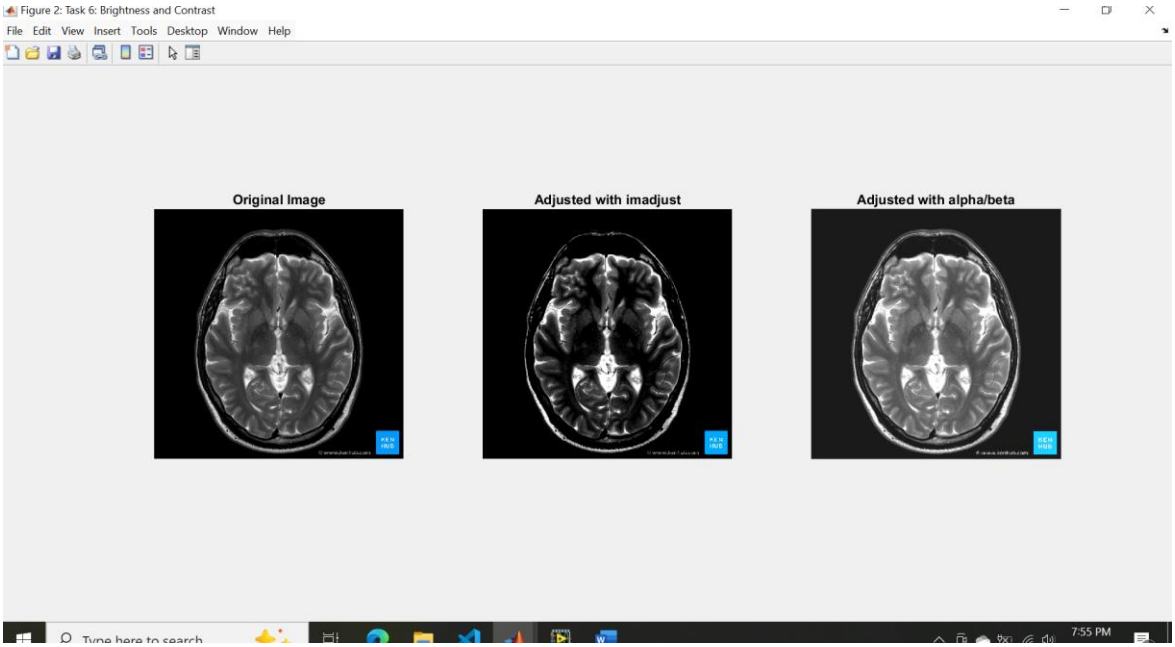


Code:

```
% ========== READ THE IMAGE ==========
imageFileTask6 = 'image2.jpg';
I6 = imread(imageFileTask6);

% ========== ADJUST BRIGHTNESS & CONTRAST ==========

% ----- Option A: Using imadjust directly -----
% imadjust(I, [low_in, high_in], [low_out, high_out], gamma)
% Example: Increase contrast by mapping [0.2,0.8] -> [0,1], no gamma change
I6_adjusted_imadjust = imadjust(I6, [0.2 0.8], [0 1], 1);

% ----- Option B: Using immultiply & imadd -----
% alpha = contrast factor; beta = brightness offset (in [0,1] if double)
alpha = 1.2;  % >1 = higher contrast
beta  = 0.1;  % positive => brighter
```

```
% Convert to double in [0,1]
I6_double = im2double(I6);

% Scale intensities (contrast), then add offset (brightness)
I6_temp = immultiply(I6_double, alpha);
I6_temp = imadd(I6_temp, beta);

% Clip values back to [0,1], then to uint8 if desired
I6_temp = max(min(I6_temp, 1), 0);
I6_adjusted_manual = im2uint8(I6_temp);

% ========== DISPLAY RESULTS ==========

figure('Name','Task 6: Brightness and Contrast');
subplot(1,3,1);
imshow(I6);
title('Original Image');

subplot(1,3,2);
imshow(I6_adjusted_imadjust);
title('Adjusted with imadjust');

subplot(1,3,3);
imshow(I6_adjusted_manual);
title('Adjusted with alpha/beta');
```

Output:



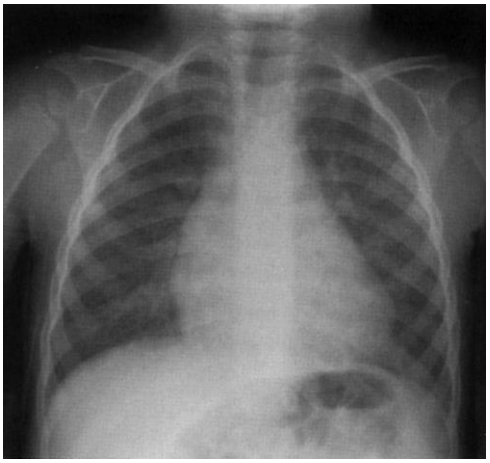
Explanation:

Brightness typically involves adding an offset (β) to all pixel values, shifting the overall intensity up or down. *Contrast* generally involves scaling (multiplying by α), pushing dark and light values further apart if α>1.

In MATLAB, we can do this manually using immultiply (for contrast) and imadd (for brightness), or with imadjust by mapping a chosen input range to a desired output range. We did both ways and the results are shown above.

Task 7: Parameter Exploration and Results Analysis

Image used:

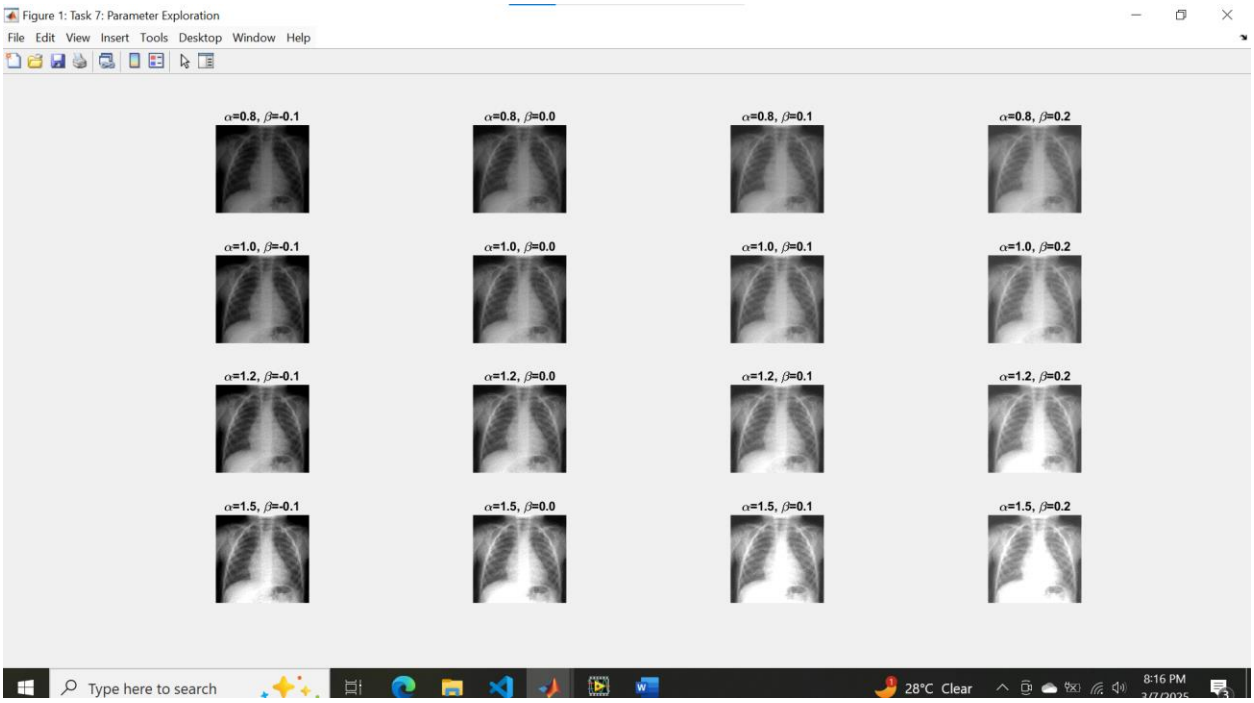


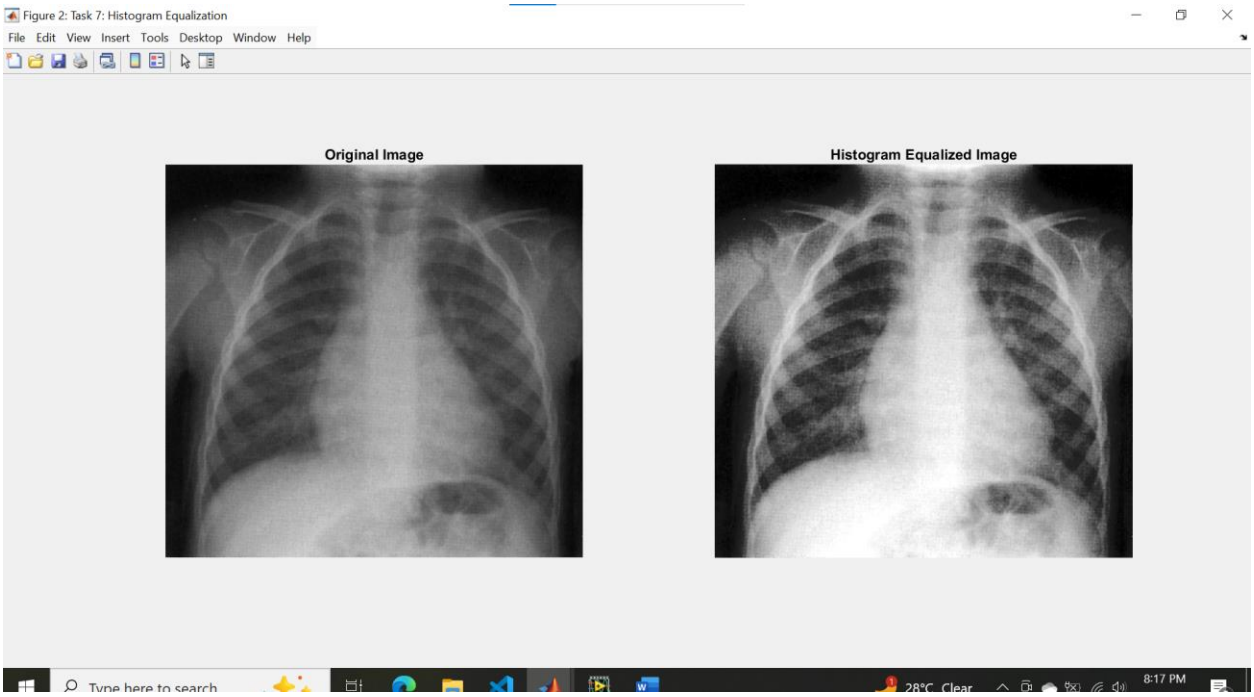
Code:

```
imageFileTask7 = 'image3.jpg';
I7 = imread(imageFileTask7);

% === EXPLORING BRIGHTNESS & CONTRAST PARAMETERS ===

alphaValues = [0.8, 1.0, 1.2, 1.5];  % Contrast factors
betaValues  = [-0.1, 0, 0.1, 0.2];   % Brightness offsets (in [0,1] if double)

figure('Name','Task 7: Parameter Exploration');
idx = 1;
for a = 1:length(alphaValues)
   for b = 1:length(betaValues)
      % Convert to double [0,1]
      tempDouble = im2double(I7);

      % Apply alpha (contrast) and beta (brightness)
      tempDouble = immultiply(tempDouble, alphaValues(a));
      tempDouble = imadd(tempDouble, betaValues(b));
```

```
        % Clip values to [0,1]
        tempDouble = max(min(tempDouble, 1), 0);

        % Convert back to uint8
        tempUint8 = im2uint8(tempDouble);

        % Display in a grid of subplots
        subplot(length(alphaValues), length(betaValues), idx);
        imshow(tempUint8);
        title(sprintf('\\alpha=%.1f, \\beta=%.1f', alphaValues(a), betaValues(b)));
        idx = idx + 1;
    end
end

% === HISTOGRAM EQUALIZATION ===
% we can also compare standard histogram equalization:
I7_histeq = histeq(I7);

figure('Name','Task 7: Histogram Equalization');
subplot(1,2,1);
imshow(I7);
title('Original Image');

subplot(1,2,2);
imshow(I7_histeq);
title('Histogram Equalized Image');
```

Output (1) Parameter Exploration (2) Histogram Equalization:





Explanation and Observations:

Parameter Exploration

- We varied α and β to see how each combination affects image appearance. In biomedical images, for instance, we might be looking for settings that highlight certain tissues or anatomical details without washing them out.

Histogram Equalization

- histeq(I) is MATLAB's built-in function for histogram equalization, which redistributes pixel intensities so that the histogram of the output image is (approximately) uniform. This increases global contrast.
- We have to be aware of the over-equalization that can emphasize noise in flat regions and that is a tradeoff we encounter while performing Image processing. We might prefer more localized techniques like adapthisteq (Contrast Limited Adaptive Histogram Equalization) for finer control.