General Instructions

Due Date: Sunday, November 20th, 2022 by 11:59pm (submit via Zybooks)

Assignment Instructions:
This assignment has two problems, summarized below. You will use MATLAB as a tool to solve the problem for the given test cases, ensuring that your code is flexible for any additional test cases that might be used to evaluate it.

- MRI Scanner (Application to Biomedical Engineering)
- Photo Editing Tools (Application to Image Processing and Software Development)

Zybooks Submission Instructions:
After completing this assignment in MATLAB, to receive credit, you must submit your code in Zybooks. The following components must be submitted under the specified chapter of the course Zybooks before the deadline to receive credit.

- Chapter 37.1 MA8 Problem 1: Main Script
- Chapter 37.2 MA8 Problem 2: LightDark Function
- Chapter 37.3 MA8 Problem 2: Grayscale Function
- Chapter 37.4 MA8 Problem 2: Colorblind Function
- Chapter 37.5 MA8 Problem 2: Main Script

To submit your script, copy and paste your code into the submission window, making sure to remove any housekeeping commands. You may submit to Zybooks as many times as you want before the deadline, without any penalty. The highest score attained before the deadline will be graded. All components are due before the due date. No credit will be given if it is not submitted through the Zybooks platform before the deadline. Credit for each component will be awarded based upon the percentage of successfully completed assessments.

Proficiency Time: Times are included with the Background and Task sections. These times are the estimated amount of time it should take you to **redo** an assignment once you are fully proficient in material that it covers. To practice, reread the background in the given Comprehension Time and attempt to complete the problem in the given Proficiency Time.

Academic Honesty Reminder

The work you submit for this assignment should be your work alone. You are encouraged to support one another through collaboration in brainstorming approaches to the problem and troubleshooting. In this capacity, you are permitted to view other students' solutions, however, copying of another student's work is strongly discouraged.

This assignment will be checked for similarity using a MATLAB code. The similarity code will check each submission for likeness between other student submissions, past student submissions, the solution manual, and online resources and postings. If your submission is flagged for an unreasonably high level of similarity, it will be reviewed by the ENGI 1331 faculty, and action will be taken by faculty if deemed appropriate.

NOTE: Since this is an automated system for all sections, if any of your work is not your own, you will be caught. Changing variable names, adding comments, or spacing will not trick the similarity algorithm.

Problem 1: MRI Scanner

Background: Comprehension Time: 5 min

Hospitals and other medical facilities frequently must use magnetic resonance imaging (MRI) scanners to create images of the human body and identify areas of disease or health. MRIs utilize magnetic fields and radiofrequency radiation to create detailed, cross-sectional, grayscale profiles of organs that can then be analyzed by medical personnel. A common use of MRIs is to identify cancerous tumors in the human body.

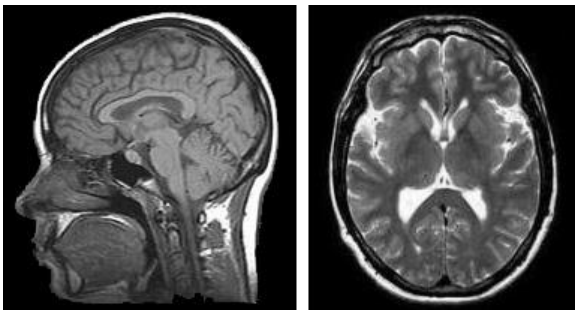


Figure 1: Example of an MRI of the Human Brain

You are working as a biomedical engineer for a cancer treatment center and wish to automate the process of identifying brain tumors in patients to reduce the workload of other medical staff. Your program, given an image of the brain tumor, must efficiently locate the tumor in an MRI scan of a patient's brain. You have been provided two examples to test with.

NOTE:

To ensure that your program is compatible with the MRI Scanner, **you can only use** the following functions:

length(), size(), imread(), image(), error(), fprintf(), double()

You also **cannot** use implicit loops in your program. An implicit loop is defined as using MATLAB's built-in functions to index or manipulate data in an array. For example, adding 7 to the first five values of A by saying `A(1:5) = A(1:5) + 7;` instead of writing a for loop such as `for i = 1:5, A(i) = A(i) + 7; end` The first example, which uses the colon **:** as an operator, is considered an implicit loop because MATLAB is executing the second example, an explicit loop, in the background.

Tasks — Proficiency Time: 35 – 50 min

TASK 1: Data validation (5 min)
Load in the images **MRI_Scan.png** and **Tumor.png**. Check that the image of the MRI scan is larger than the image of the tumor (both rows and columns). Also check that both images are viewable images (three-dimensional, three-layered matrices). If either of these conditions are not met, produce an error and terminate the program.

TASK 2: Find the tumor (20 – 30 min)
Write a script which finds a brain tumor in a given MRI scan. When the brain tumor is found, output to the command window the pixel coordinates that the tumor is between. If the brain tumor is not found, output to the command window that the brain tumor cannot be found on this MRI scan. Your code should find the tumor, or determine the tumor is not on the MRI scan, in no more than thirty seconds of runtime.

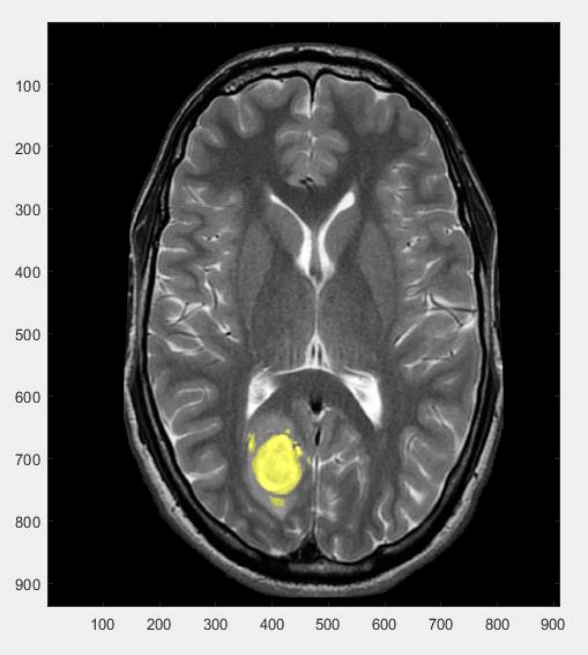
NOTE: Determining where the tumor is on the MRI scan will require the entire tumor image to be checked. Checking only a portion of the image may lead the program to output a false positive.

TASK 3: Highlight the tumor (10 – 15 min)
If the brain tumor is found, display the MRI scan with the site of the brain tumor highlighted in yellow. To determine which portion of the tumor image is to be highlighted, assume that the tumor will always be within the region of the tumor image that is brighter than the average brightness of the tumor image. To highlight the tumor, increase the pixel values in the tumor's red and green layers by 25% and decrease the pixel values in the tumor's blue layer by 50%.

HINT: Pay attention to the data type of the image when determining average brightness.

Sample Output:

```
Command Window
The tumor is located between the coordinates [640,352]
and [777,465] starting from the top left corner of the MRI scan.
>>
```



Problem 2: Photo Editing Tools

Background: Comprehension Time: 5 min

One practical application of scripting and programming languages is the development of image processing tools to allow other users to manipulate an image with little to no programming experience. Photoshop is programmed entirely in C++, and Apple's Photos software is programmed using their proprietary language Swift. Unfortunately, you are enrolled in a computing course and must program your own photo-editing tools yourself.

You must develop the following photo-editing tools described in the Tasks below by creating user-defined functions and test their effectiveness using the provided **ColorWheel.png**. Each tool you develop will be independent of the others; that is, the modifications made to **ColorWheel.png** for each Task should not all be applied to the same image. You can use additional images in a .png or .jpg format to further test your tools, if you wish.

For extra practice, attempt this problem with the following additional restrictions (optional):

These tools must be translatable to other programming languages. Because of this, **you can only use** the following functions:

length(), size(), imread(), imshow(), double(), subplot(), figure(), title()

You also **cannot** use implicit loops in your program. An implicit loop is defined as using MATLAB's built-in functions to index or manipulate data in an array. For example, adding 7 to the first five values of A by saying `A(1:5) = A(1:5) + 7;` instead of writing a for loop such as `for i = 1:5, A(i) = A(i) + 7; end` The first example, which uses the colon **:** as an operator, is considered an implicit loop because MATLAB is executing the second example, an explicit loop, in the background.

Tasks Proficiency Time (with explicit loops only): 25 – 40 min

TASK 1: Swap Brightest and Darkest Colors (15 – 20 min)
Create a function **LightDark** which identifies the brightest color of a given image (closest to white) and swaps that color with the darkest color of the image (closest to black).

HINT: The brightness of a color is proportional to the sum of its RGB values.

TASK 2: Grayscale (3 – 8 min)
Create a function **Grayscale** which converts the given image into a grayscale image. A grayscale image is a single-layer matrix which represents the luminance (brightness) of the given image. The term "grayscale" denotes that to represent this single-layer matrix as an image, said image would be composed of three identical layers, i.e. all shades of true gray. The formula to convert an image to grayscale is

$$\text{luminance} = 0.2989*\text{red} + 0.5870*\text{green} + 0.1141*\text{blue}$$

TASK 3: Deuteranomaly (2 – 7 min)
Suppose you wish to understand how much the colors of the image are distinguishable to someone who suffers from deuteranomaly (red-green colorblindness). Create a function **Colorblind** which applies a filter to model how a deuteranope might see the image. A deuteranomaly filter can be developed by applying the formulae

$$\text{red} = \text{red}*0.66 + \text{green}*0.34$$
$$\text{green} = \text{red}*0.55 + \text{green}*0.45$$
$$\text{blue} = \text{green}*0.25 + \text{blue}*0.75$$

TASK 4: Applications (5 min)

Load the image **ColorWheel.png**. Apply the three tools developed above (by calling the functions) to three separate copies of the given image. Display the original image with the three modified images together in two ways:

- As four separate images on one figure, with appropriate titles
- As one image arranged together on another figure, with an appropriate title

Sample Output: