**DIP Assignment 1**

**Muhammad Moiz Sajid**

**Roll Number: 19i-0692**

**Section: B**

**Question 1:**

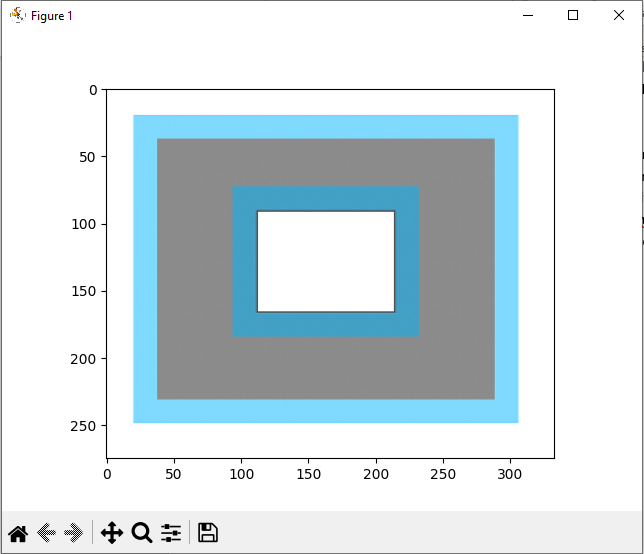
* Pythons OpenCV has been used to open and read the image files i.e **“Test.png”**
* cv2.imread() opens the file in a **“cv2.IMREAD\_UNCHANGED”** to open the images in ARGB mode.
* Note: OpenCV opens the image in a BGR format.
* A print statement at the start is used to check the transparency.
* **“img.shape”** returns the rows and columns of the image respectively which are then checked for the range **80 to 85** for gray colour and if found, replaces its transparency.
* Finally the image is displayed using matplotlib’s **“imshow()”** function.
* The colours might be inverted because matplot reads the image in RGB while opencv reads in BGR format.

**Input:**

**A picture containing qr code

Description automatically generated**

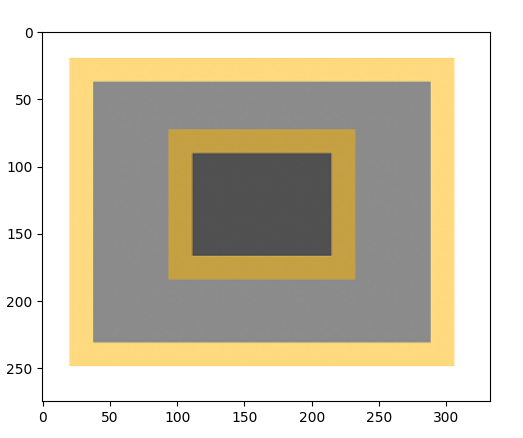
**Output:**



**Question 2:**

* Matplot’s **“imread()”** has been used to open the image.
* A function **“ConvertToGray()”** has been created.
* The function essentially slices the image into respective Red, Green, and Blue colours for each index and applies the given formula for multiplication of each RGB colour with the respective index of multiplication.
* Finally the image is displayed using matplotlib’s **“imshow()”** function.

**Input:**

****

**Output: A picture containing qr code

Description automatically generated**

**Question 3:**

* Python’s **Pillow library** has been used to display and manipulate the images.
* Two images are loaded using the **“Image.open()”** function.
* Sizes of the images are compared using **“Image.size”** function and used to make a check before concatenating images.
* **“Image.new()”** has been used to create a new array of the size 758x277, which has double the horizontal pixels for the purpose of holding a concatenated image.
* Both the images are concatenated in the **“newImage”** using the **“Image.paste()”** function.
* The new formed image is shown using the **“Image.show()”** function of pillow.

**Input:**

**A white rectangle with a black background

Description automatically generated with low confidence**

**Output:**

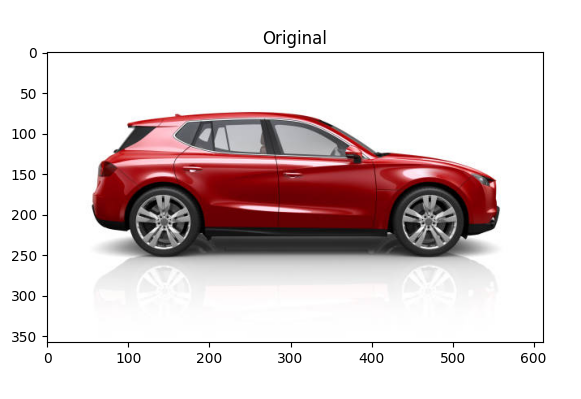
**A picture containing logo

Description automatically generated**

**Question 4:**

* Python’s **Pillow library** has been used to display and manipulate the images.
* The image is loaded using the **“Image.open()”** function. A new image **“car.jpg”** has been imported and used for the purpose of this question.
* **Pillow library’s transpose()** function has been used for vertical and horizontal flipping.
* The transpose function’s **Image.FLIP\_TOP\_BOTTOM** attribute has been used for vertical flip whereas, the **Image.FLIP\_LEFT\_RIGHT** attribute has been used for horizontal flip.
* Finally the images are displayed using matplotlib’s **“imshow()”** function.

**Input:**

****

**Output:**

A picture containing chart

Description automatically generated

A picture containing diagram

Description automatically generated

**Question 5:**

* Pythons OpenCV has been used to open and read the image files.
* **cv2.imread()** opens the file in a **“cv2.IMREAD\_GRAYSCALE”** to open the images in **grayscale mode**.
* Numpy’s **“zeros()”** method is used to create a **2d-np array** of size **277x379** to store the results of the common area between the two images.
* **“img.shape”** returns the rows and columns of the images respectively which are then used to check for the common between the two images.
* If such areas are found, we store the newly formed array **“newImg”**.
* Finally, opencv’s **“imshow()”** method is used to show the resultant image array.

**Input:**

Rect1:

**Shape

Description automatically generated with low confidence**

Rect7:

**Logo

Description automatically generated**

**Output:**

**A picture containing graphical user interface

Description automatically generated**