Assignment 3

Prepared By:

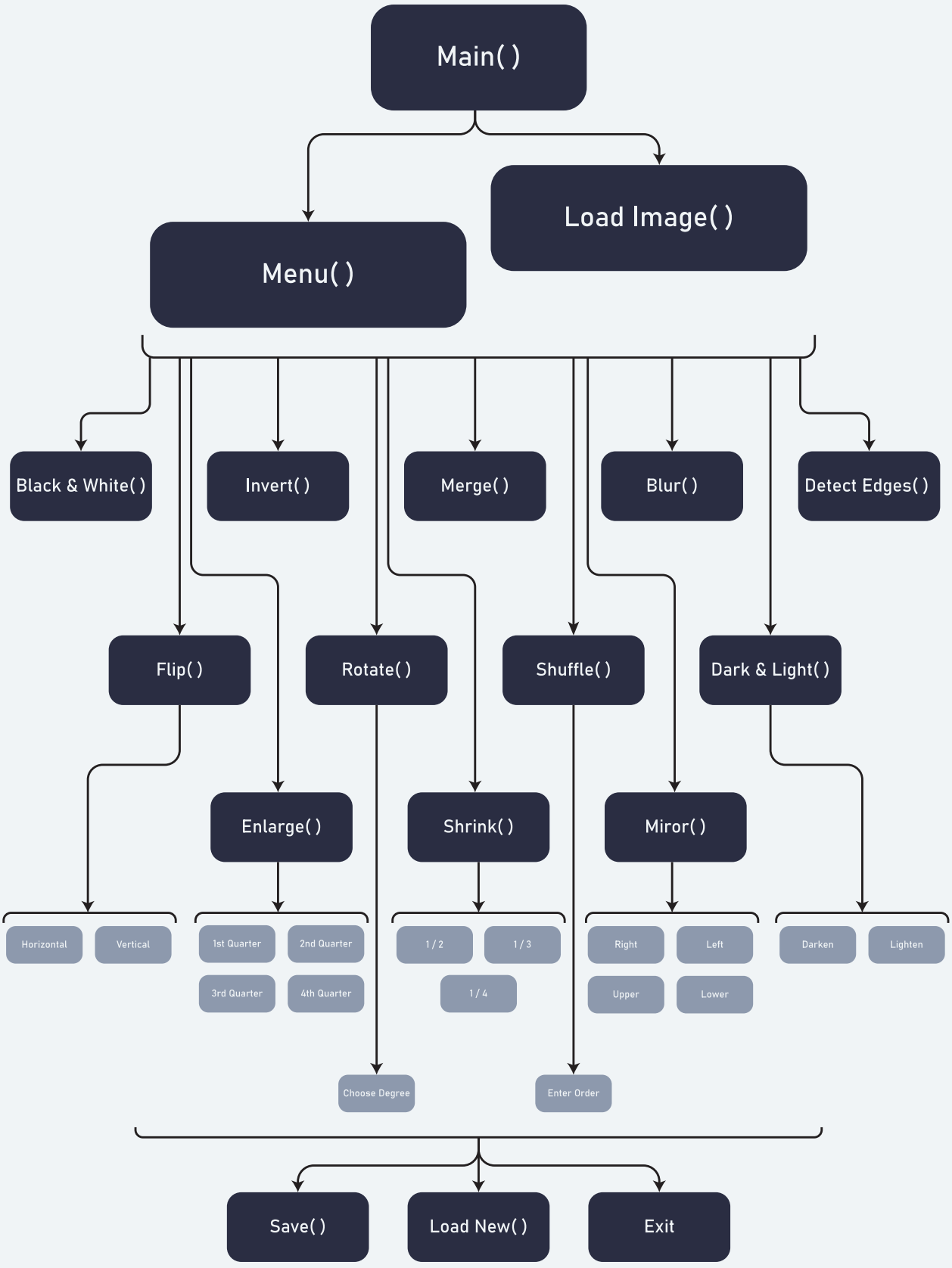
Mohamed Ehab Tawfik	20210331
Ahmed Reda El-Sayed	20210018
Salma Mohammed Mahmoud	20210161

S7,8

Under Supervision of:

Dr. Mohammed El-Ramly

System Diagram



1. Black and White Image

If you apply this function to the loaded image, it will produce another version of the image that is black and white. You can do this by calculating the average gray level for all pixels in the image. And then every pixel above the average is turned to white (255) and every pixel below average is turned to black (0).

Algorithm:

Step 1: Start

Step 2: Load Image

Step 3: Calculate the average of the gray in the whole image

Step 4: Loop on each pixel in the image

Step 5: Compare the pixel value to the average of the gray

Step 6: If the pixel is bigger than the average makes it white

Step 7: Else: make it black

Step 8: Save Image

Step 9: End

2. Invert Image

If you apply this function to the loaded image, it will produce the negative of the image and you can store it in the file name you give. The negative has every black pixel turned to white and every white pixel turned to black, and every gray pixel is turned to opposite level of brightness (255 – pixel value).

Algorithm:

Step 1: Start

Step 2: Load Image

Step 3: Loop for each pixel in the image

Step 4: Make each pixel = 255 - pixel

Step 5: Save Image

Step 6: End

3. Merge Images

In this function, you will be asked to enter the name of another image. Then the program will load this image. The program will create a new image, with every pixel equal the average gray level of the corresponding pixels in the images to merge.

Algorithm:

Step 1: Start

Step 2: Load 2 Images

Step 3: Loop for each pixel in the first image and the second image

Step 4: Get the average of every 2 pixels at the same place

Step 5: Replace the pixel with the new value (average)

Step 6: Save Image

Step 6: End

4. Flip Image

This filter allows the user to flip the image horizontally or vertically, as if it is reflected on a mirror.

Algorithm:

Step 1: Start

Step 2: Load Image

Step 3: Output “do you want to flip the photo horizontally or vertically”

Step 4: Input choice

Step 5: If (choice==horizontal) {

Step 6: Loop of each pixel in the image

Step 7: Store the first row in the image in the last row in another image and store the second row in the penultimate row and so on
}

Step 8: If (choice == vertical) {

Step 9: Loop of each pixel in the image

Step 10: Store the first column in the image in the last column in another image and store the second column in the penultimate column and so on
}

Step 11: Save Image

Step 12: End

5. Darken and Lighten Image

This filter allows the user to make the image darker or lighter by 50%.

Algorithm:

Step 1: Start

Step 2: load image

Step 3: user input darken or lighten

Step 4: if user choose darken

Step 5: make two nested for loop to divide each pixel in 2D array by (2) to darken by 50%

Step 6: else if user choose lighten

Step 7: make two nested for loop to add (pixel / 2) for each pixel in 2D array to lighten by 50%

Step 8: but before change any pixel check that the number after change will not be more than 255

Step 9: if more than 255, then pixel = 255

Step10: else, add (pixel / 2) for each pixel in 2D array

Step11: save image

Step12: End

6. Rotate Image

This filter allows the user to rotate the image clockwise by 90°, 180° or 270° as the user chooses.

Algorithm:

Step 1: Start

Step 2: Load Image

Step 3: Ask the user about the degree (90 – 180 – 270)

Step 4: If the degree is 90

Step 5: Store the pixel in the row (column number) and the column (255- row number)

Step 6: If the degree is 180

Step 7: Store the pixel in the row (255 – row number) and the column (255 – column number)

Step 8: If the degree is 270

Step 9: Store the pixel in the row (255 – column number) and the column (row number)

Step 10: Save Image

Step 11: End

7. Detect Image Edges

This function finds the edges of the drawings in the image and turns the image into a skeleton version of the original as if it is drawn with pencil without coloring as shown.

Algorithm:

Step 1: Start

Step 2: Load Image

Step 3: make two nested for loop

Step 4: inside the two for loop subtract the pixel from upper pixel once and from behind pixel once

Step 5: take the absolute as we don't need negative number

Step 6: then check if the result more than or equal 40

Step 7: if more than or equal 40, pixel = 0

Step 8: else, pixel = 255

Step 9: Save Image

Step 10: End

8. Enlarge Image

This filter allows the user to enlarge one of the four quarters of the image into a separate new image.

Algorithm:

Step 1: Start

Step 2: Load Image

Step 3: Input quarter you want to enlarge

Step 4: If (quarter==1) {

Step 5: Loop for each pixel in first quarter image from image [0][0] to image [127][127]

Step 6: Store this pixel in another image in four pixels starting from image [0][0]}

Step 7: If (quarter==2) {

Step 8: Loop for each pixel in second quarter image from image [0][127] to image [127][255]

Step 9: Store this pixel in another image in four pixels starting from image [0][0]}

Step 10: If (quarter==3){

Step 11: Loop for each pixel in third quarter image from image [127][0] to image [255][127]

Step 12: Store this pixel in another image in four pixels starting from image [0][0]}

Step 13: If (quarter==4){

Step 14: Loop for each pixel in fourth quarter image from image [127][127] to image [255][255]

Step 15: Store this pixel in another image in four pixels starting from image [0][0]}

Step 16: Save Image

Step 17: End

9. Shrink Image

This filter allows the user to shrink the image dimensions to 1/2, 1/3 or 1/4 the original dimensions.

Algorithm:

Step 1: Start

Step 2: Load Image

Step 3: user input (1/2 – 1/3 – 1/4)

Step 4: make two nested for loop4

Step 5: if user choose 1/2, each two pixels will be store in one pixel to shrink the image to half

Step 6: else if user choose 1/3, each three pixels will be store in one pixel to shrink the image to third

Step 7: else if user choose 1/4, each four pixels will be store in one pixel to shrink the image to fourth

Step 8: Save Image

Step 9: End

a. Mirror Image

This filter mirrors 1/2 of the image as seen here in order: Left 1/2, Right 1/2, Upper 1/2 and Lower 1/2.

Algorithm:

Step 1: Start

Step 2: Load Image

Step 3: Input the side that you want to mirror it

Step 4: If (side==left) {

Step 5: Loop for each pixel in left side starting from image [0][0] to image [255][127]

Step 6: Store each pixel in the right side of the image}

Step 7: If (side==right) {

Step 8: Loop for each pixel in right side starting from image [0][127] to image[255][255]

Step 9: Store each pixel in the left side of the image}

Step 10: If (side==upper) {

Step 11: Loop for each pixel in the upper side starting from image [0][0] to image[127][255]

Step 12: Store each pixel in the lower side of the image}

Step 13: If (side==lower) {

Step 14: Loop for each pixel in the lower side starting from image [127][0] to image[255][255]

Step 15: Store each pixel in the upper side of the image}

Step 16: Save Image

Step 17: End

b. Shuffle Image

Assume the image consist of 4 quarters as shown, the user enters the order he wants to the quarters in the new image. Wrong input is rejected. Assume he entered 4 3 2 1 he gets image 1. Or if he enters 3 1 4 2 he gets image 2. User can enter any order he likes.

Algorithm:

Step 1: Start

Step 2: Load Image

Step 3: save each quarter in new 2D array

Step 4: user input the order to rearrange the image

Step 5: make two nested for loop, but not for loop in all 2D array for loop on quarters only

Step 6: as each quarter saved in 2D array, the first quarter in the image will be equal to the first quarter in the order that the user input

Step 7: the second quarter in the image will be equal to the second quarter in the order that the user input

Step 8: the third quarter in the image will be equal to the third quarter in the order that the user input

Step 9: the fourth quarter in the image will be equal to the fourth quarter in the order that the user input

Step 10: Save Image

Step 11: End

C. Blur Images

This filter produces a bury version of the image.

Algorithm:

Step 1: Start

Step 2: Load Image

Step 3: Loop on each pixel in the image

Step 4: Calculate the average of the pixel with the 8 pixels surrounding it (Kernel Algorithm)

Step 5: Put the average in this pixel place

Step 6: If the pixel is on the edges, we replace the missing pixels with the value of the average

Step 7: Save Image

Step 8: End