

**Title:** Assignment : Module 1

**Name:** Umm-e-Hani

**Roll No.:** 21I-1715

**Section:** M

**Submitted to:**

M. Ishtiaq

TABLE OF CONTENT

| **Sr. No** | **Content** |
| --- | --- |
| **1.** | Introduction |
| **2.** | Image class |
| **3.** | Question # 1 |
| **4.** | Question # 2 |
| **5.** | Question # 3 |

# 

# **INTRODUCTION:**

The assignment is based on working with the binary images using abstract data types. Before the questions, it is required to form an image class with some basic functions to use throughout the assignment. In the assignment, it's also required to

To implement connected component extraction using stack and queue.

**IMAGE CLASS:**

The image class consists of the following functions:

* Read(string)
* Save()
* SetPixel(int,int,int)
* GetPixel(int, int)
* GetSize()
* ConvertToNegative()

**IMPLEMENTATION OF THE FUNCTIONS:**

The read function was the most hard to implement out of all the functions as the code that I implemented was only appending the pixel value 0 into the dynamic 2D array. In the read function, I used a static string array of size 6 that stored the starting 6 lines of the pgm file. I created a sstream object to get rows and columns. The error I faced was from here because I was reading the array values from the 6th line of the document and it wasn’t appending the value 255 into the array. It took me some time to figure out my mistake. I corrected it and the code was working again. The pixel values were appended to the array and then the file is closed at the end.

The other functions save, get pixel , set pixel, convert to negative were easy to implement once the read function was correctly coded. The most challenging part was the read function in this class. The other functions were coded without any error or challenge.

**QUESTION # 1**

In question 1, the following functions were implemented:

* Mean()
* numPixels()
* avgBlackpixels()
* ConvertToNegative()

The mean function returns the mean of the pixel values in the array. All the values of the array are summed and stored in a variable called mean which is then divided by the total size of the image. The numPixel function displays the number of black and white pixels. The pixel value 0 is considered black and 255 is considered white. The avgBlackpixels calculated the number of average black pixels in a row and convert to negative function to change the black pixels i.e 0 to white pixels i.e 255 and vice versa.

**QUESTION # 2**

Question 2 was quite challenging to do at first, I had to implement a template queue. The template queue was really hard to implement as I had to then use two template classes: the node class and the queue class. I kept getting the error about the node class and the constructor. I had to search around alot about the error and warnings I was getting. I finally found the solution to it which was to use the node class as a template in the class definition. The queue was finally implemented. Then regarding the actual requirement in the question I read the links provided which were quite hard to understand. I searched about connected components on youtube from which i got to know that the pixel location that the user will provide we have to check its neighbors. A pixel has 8 neighbors in total. The way to check them was simple by adding or subtracting in the x-axis and y-axis according to the neighbor we wanna check. Hence 8 checks were used in total. The approach that I used was to create a temporary array which was initialized by 255 at every index. Then I created a queue object and enqueued the x and y location axis given by the user. There is a while loop which will continue until the queue is not empty. The following 8 if checks are then coded in the while loop for the neighbors. My code didn't work at first the output was a white blank image. My error was that when I was dequeueing the values I placed the x location in y and vice versa. I tracked that problem down and corrected it along with it. I had to add the index bound check as well. Also had to add the array value check to 255 as well the the temp array was set to 0 at that location. Hence after some errors from the queue at the start changing it the rest was easy to implement.The function in which the code was implemented is:

* imageQueue(int,int)

# **QUESTION # 3**

Question 3 was implemented the same as Question 2 just that instead of using the queue class, stack class was used. And the push , pop operations were used instead of enqueue and dequeue. I also observed that the program executed much faster with stack instead of queue. The function in which the stack code was implemented is:

* imageStack(int,int)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_END OF REPORT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**