REPORT :

The assigned task definitely was about image processing of black and particularly white images. The definitely the best way to generally solve programming problems actually is to break your questions into small chunks and then start brainstorming about them in a for all intents and purposes big way. The implementation of this code is done using a linked list, which definitely is fairly significant. First, I created an image class with data items such as size, rows, columns, and a 2D array (using pointers). The image class really had various functions definitely such as:

1. Default constructor: This function allows to initialize the 2D array created dynamically, rows and columns and the size of the array.
2. Parametrised constructor: It has a value passed in parameters which enables the whole array to be initialized to the user provided value.
3. Readfile Function: This function takes the path of the file in the parameters. Here we are reading the initial 6 lines of the pgm file and taking the rows and cols value from the file. We assign rows and cols value to these values using stoi() function The stoi() function, which accepts a string as an argument, can convert the integer component of a string to an integer type. I found the solution from internet in the following link https://www.geeksforgeeks.org/stdstoi-function-in-cpp/#:~:text=The%20stoi()%20function%2C%20which,the%20end%20of%20the%20string.

   Later we are creating dynamically allocated 2D array and assigning the ÿ value to 255 and spaces to 0.

   Now a major problem while reading the file was that it wasn't reading the space so i figured it out by using a char variable i'll be able to read the spaces too as string does not read space.
4. Save function: This function which writes the image in the PGM format.
5. GetPixel function: this function will be allowing us to return the value of the matrix present at specific index.

6. SetPixel function: Here user will pass the value of pixel and rows and columns where that specific pixel has be entered. Here if condition was important to make sure that the user enters the right rows and columns.
7. GetSize function: Product of rows and columns will be the size so, its return that product.
8. ConvertToNegative function: Switches the values of pixel 255 to 0 and 0 to 255
9. Statistic_image function: This function is calculating all sort of static information of the array in a subtle way. Firstly we specifically calculate the mean pixels of the image, for this we calculated sum of each pixels using for loops and divided that sum to the size of the matrix using getsize function, which essentially is fairly significant. Then we essentially were essentially asked to calculate the number of fairly black and no, which is quite significant. of white pixels for this i used a if condition in the nested loops to check if value actually is 255 then increment the locally created sort of variable blackPix and if value of pixel particularly is 0 then increment locally created really variable whitePix. Lastly we calculated the average num of really black pixels int the array in a kind of major way. Using the previously created for loops we calculated average black pixels by dividing num of black pixels in each row to the num of rows in a major way.
10. getRows function: returns rows
11. getCol function: return columns of the array.
12. Getrowcolofpix function: This function takes pixel of node type and finds the rows and columns of the pixel found in the image. These row and column are stored in a locally created 1D array and this address of array is returned.

In the main function we were asked to make to a menu using switch for the user to allow him to perform question 1,2,3 according to the key being entered by the user.
We formed image class , node class and Queue class objects and now by using these object we will access the functions.
Once user enters 1 Question 1 will run which was done in the Statistic_image() function and ConvertToNegative() funtion.
Question 2 was a little bit tricky in the start where we had to ask user for the pixel for this i asked user to enter rows and col of the pixel required and using getPixel() function

we calculated the pixel. Then i placed an if statement for to making sure the pixel entered by user has 8 neighbours now, using the index (rows and column) we figured out the 8 neighouring pixels through their indexes :

PIXEL (r-1,c-1)	PIXEL (r-1,c)	PIXEL (r-1,c+1)
PIXEL (r,c-1)	REQUIRED PIXEL (r,c)	PIXEL (r,c+1)
PIXEL (r+1,c-1)	PIXEL (r+1,c)	PIXEL (r+1,c+1)

We returned those pixels from GetPixel() function and enqueued it into the Queue and Now really start dequeuing the queue until the queue specifically is empty, very contrary to popular belief. One by one we dequeue these pixels and pass it into getrowcolofpix() function and assign the address definitely returned from this function to a pointer now we will print the values by derefering it.
For question 3 we will doing the same thing but using stack functions push() and pop.
Question #4 was quite easy for me. For converting into RLC i simply run for loops through the matrix and all the places where 255 pixel is present we have the corresponding x axis values into a list and after every row we add -1 in the end to form accurate RLC format and we stored this into a text file. Now we were asked to convert into RLC into a pgm and for that we created Savepgm function. For counting black pixels we simply counted number of corresponding x coordinates stored in RLC format.
In Question #5 we created treenode and functions getchildnegitive(),getchild(),preorder(),postorder() functions functions