IMAGE PROCESSING USING C++

A PROJECT REPORT

SUBMITTED IN FULFILLMENT FOR THE ASSESMENT

OF 3RD SEMESTER

BACHELOR OF TECHNOLOGY IN

INFORMATION TECHNOLOGY

SUBMITTED BY:

Abrar Zahoor (2K19/IT/004) & Amaan Bilal(2K19/IT/011)

Under the Supervision of

Mrs. Swati Sharda,

Asst. Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

DECEMBER, 2020

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATES' DECLARATION

We, (Abrar Zahoor and Amaan Bilal), Roll No – 2K19/IT/004 and 2K19/IT/011 respectively, student of B.Tech. (INFORMATION TECHNOLOGY), hereby declare that the project Dissertation titled "Image Processing Using C++" which is submitted by us to the Department of INFORMATION TECHNOLOGY, Delhi Technological University, Delhi in fulfilment for the assessment of Discrete Structures(DST)- 3rd Semester 2020-21 , is original and not copied from any source without proper citation.

Place: Delhi
Date: 29-11-2020

Abrar Zahoor

and

Amaan Bilal

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled “Image Processing Using C++” which is submitted by Abrar Zahoor and Amaan Bilal; Roll No – 2K19/IT/004 and 2K19/IT/011 respectively; INFORMATION TECHNOLOGY, Delhi Technological University, Delhi in fulfilment for the assessment of 3rd Semester for the Course work of Discrete Structures(DST) , is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any previous assessment in any other course or degree.

Place: Delhi
Date: 29-11-2020

SWATI SHARDA
SUPERVISOR

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

We would like to further our special thanks of gratitude to our guide and Instructor for the course of DST, Ms. Swati Sharda, for her unparalleled and unflinching guidance and support for the completion of this Project.

We are immensely filled with feelings of gratitude and indebtedness for our able Professors, as they have helped us in every possible way to provide us with the knowledge required to complete the coursework and further implement the same in the domain of practical applications.

Abrar Zahoor
and
Amaan Bilal

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ABSTRACT

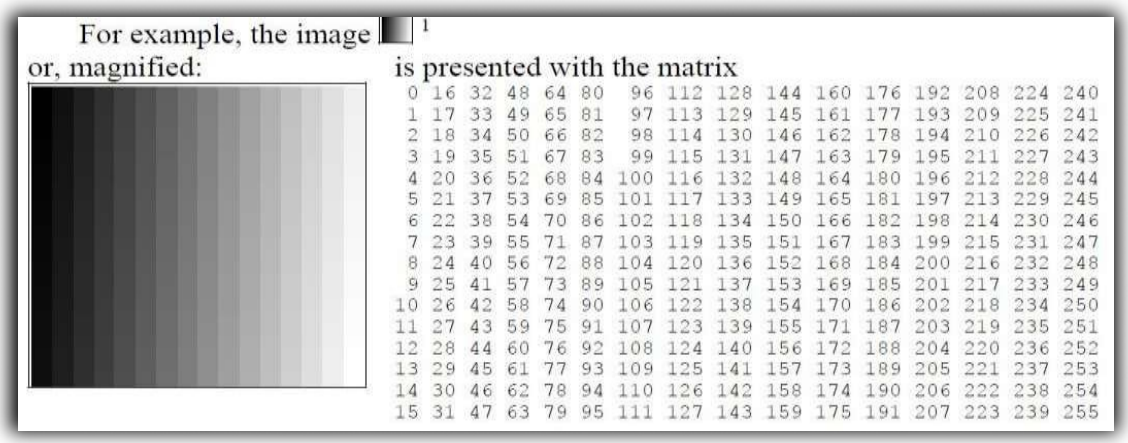
Image processing is a method or technique to convert an image into digital form and execute some operations on it, in order to acquire an enhanced image or to extract some useful information from it. Image processing also can be described as a processing of images using mathematical operations for which the input is an image such as a photograph while the output of image processing may be either an image or a set of characteristics related to the image. These techniques can be applied on images for modification or enhancement. The objectives of this Project are to define the meaning and scope of image processing by Practically implementing various Image processing Algorithms using C++.

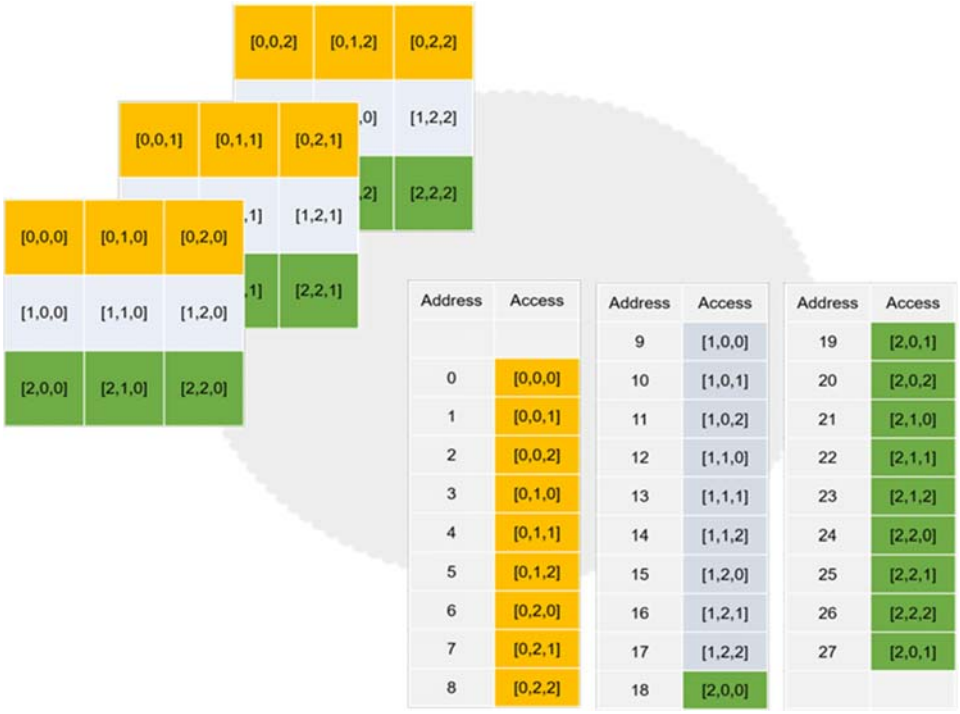
Furthermore, in this report, we discuss the various steps and methodologies involved in a typical image processing, and applications of image processing tools and processes in the frontier areas of Research.

CONTENTS:

1) IMAGE AND ITS MATRIX

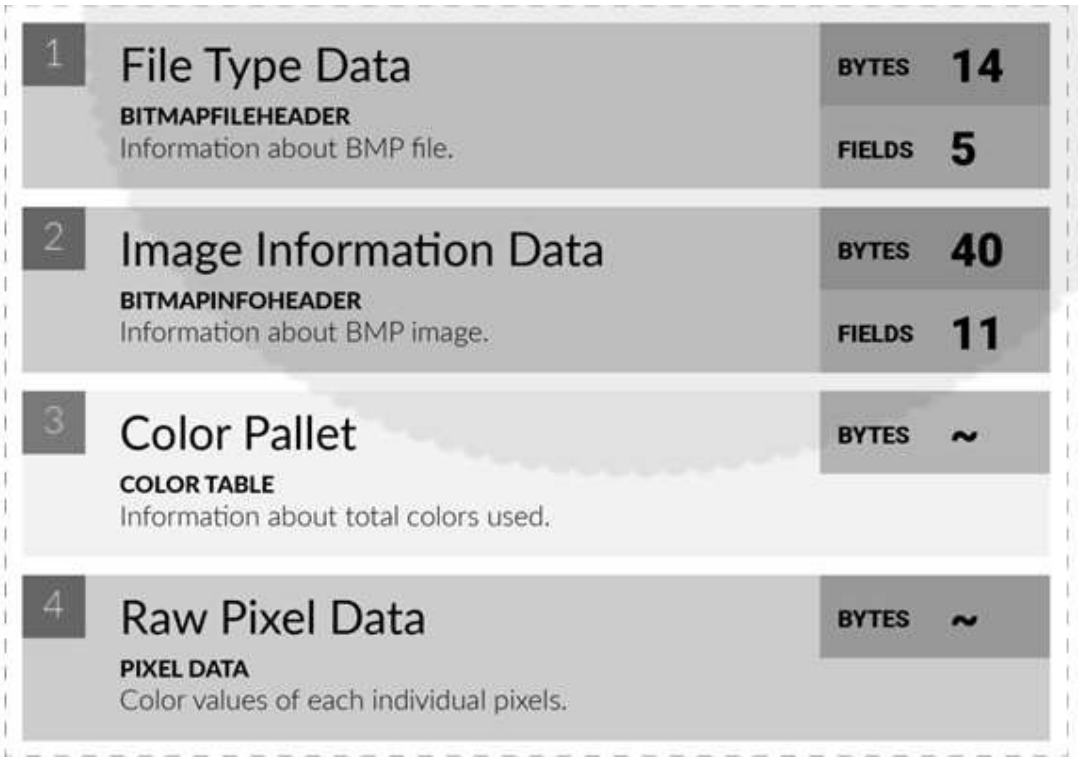
A digital grayscale image is presented in the computer by pixels matrix. Each pixel of such image is presented by one matrix element – integer from the set {0,1, 2,255}. The numeric values in pixel presentation are uniformly changed from zero (black pixels) to 255 (white pixels).
Color images (with RGB color model) in a computer are presented with three grayscale images matrices (one for each – red, green and blue – color components.





2) BMP FILE FORMAT

A bitmap image is a raster image (containing pixel data as opposed to vector images) format. Each pixel of a bitmap image is defined by a single bit or a group of bits. Hence, it is called the bitmap or a map of bits and pixels.
BMP allows encoding images in different color depths. The color depth is a measure of an individual image pixel to accurately represent a color. Color depth is calculated in bits- per-pixel or bop.
1-bit color depth or 1bpp means a pixel can have a 1-bit color or 2 values. Monochromatic images have 1-bit color depth because a pixel can be true black or true white. BMP format supports 1- bit, 2- bit, 4-bit, 16-bit, 24-bit, and 32-bit color depths.
An image file contains information other than pixels. For example, the width and height of an image (in pixels), size of the image (in bytes), bit-depth of pixels (in bop), color pallets, etc. This is called metadata.
A BMP file format contains different sections that contain information about metadata, color pallet, and actual pixel data.



CONTD.

BMP FILE FORMAT (Continued)

Block 1-File Type Data: This block is a BMP Header. This is the starting point of the BMP file and has 14 bytes width. This header contains a total of 5 fields of variable byte width.

Block 2-Image Information Data: This is a DIB Header must be used to specify the color and image information. This header is 40-bytes wide and contains a total of 11 fields of variable byte widths.

Block 3-Colour Pallet (semi-optional): This block contains the list of colors to be used by a pixel. This is an indexed table with the index starting from 0. The integer value of the pixel points to the color index in this table and that color is printed on the screen.

Block 4-Raw Pixel Data: This block contains binary numbers dedicated to representing the unique color values of each individual pixel. Depending on the bop of the BMP image, a byte can contain color values of multiple pixels or multiple bytes can be used to represent the color value of a single pixel.

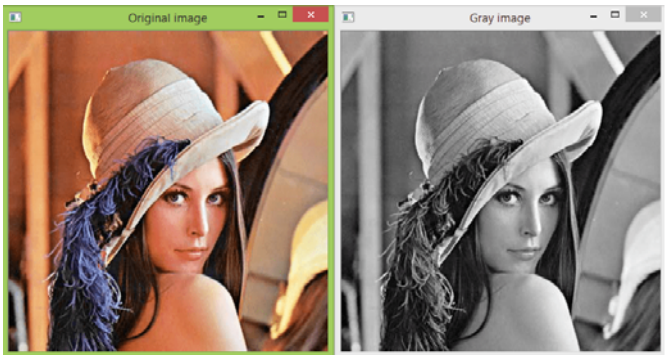
3) BMP FILE STRUCTURE:

Bitmap File Structure			
Block	**Field**	**Width**	**Description**
BITMAPFILEHEADER **Fields:** 5 **Width:** 14 bytes	FileType	2 bytes	A 2 character string value in ASCII. It must be 'BM' or '0x42 0x4D'
	FileSize	4 bytes	An integer (unsigned) representing entire file size in bytes (number of bytes in a BMP image file)
	Reserved	2 bytes	To be utilized by an image processing application. Initialized to '0' integer (unsigned) value.
	Reserved	2 bytes	To be utilized by an image processing application. Initialized to '0' integer (unsigned) value.
	PixelDataOffset	4 bytes	An integer (unsigned) representing the offset of actual pixel data in bytes.
BITMAPINFOHEADER **Fields:** 11 **Width:** 40 bytes	HeaderSize	4 bytes	An integer (unsigned) representing the size of the header in bytes. It should be '40' in decimal.
	ImageWidth	4 bytes	An integer (signed) representing the width of the final image in pixels.
	ImageHeight	4 bytes	An integer (signed) representing the height of the final image in pixels.
	Planes	2 bytes	An integer (unsigned) representing the number of color planes. Should be '1' in decimal.
	BitsPerPixel	2 bytes	An integer (unsigned) representing the number of bits a pixel takes to represent a color.
	Compression	4 bytes	An integer (unsigned) representing the value of compression to use. Should be '0' in decimal.
	ImageSize	4 bytes	An integer (unsigned) representing the final size of the compressed image. Should be '0' in decimal.
	XpixelsPerMeter	4 bytes	An integer (signed). Should be set to '0' in decimal to indicate no preference of the target device.
	YpixelsPerMeter	4 bytes	An integer (signed). Should be set to '0' in decimal to indicate no preference of the target device.
	TotalColors	4 bytes	An integer (unsigned) representing the number of colors in the color pallet.
	ImportantColors	4 bytes	An integer (unsigned) representing the number of important colors. Ignored by setting '0' in decimal.
COLOR TABLE **Fields:** 4 x entries **Width:** 4 x entries	Red	1 bytes	An integer (unsigned) representing Red color channel intensity.
	Green	1 bytes	An integer (unsigned) representing Green color channel intensity.
	Blue	1 bytes	An integer (unsigned) representing Blue color channel intensity.
	Reserved	1 bytes	An integer (unsigned) reserved for other uses. Should be set to '0' in decimal
PIXEL DATA			An array of pixel values with padding bytes. A pixel value defines the color of the pixel.

4) Image Processing Operations Implemented:

- RGB TO GREYSCALE:

When converting an RGB image to grayscale, we have to take the RGB values for each pixel and make as output a single value reflecting the brightness of that pixel. One such approach is to take the average of the contribution from each channel: (R+B+C)/3. However, since the perceived brightness is often dominated by the green component, a different, more "human-oriented", method is to take a weighted average, e.g.: 0.3R + 0.59G + 0.11B. This method is also known as “Weighted Luminosity Method”.



- RGB TO SPEIA:

Sepia tone is a reddish-brown monochrome tint. When you apply it to a photo, it gives the picture a warm, antique appearance. In the early days of photography, photos were developed using sepia, which came from the ink of cuttlefish, in the emulsion.

To convert the image into sepia, we need to apply conversion formula of RGB to sepia to all red, green and blue pixels. As the input image in our case is a rib 24-bit (8-bit for each colour), hence we know that the pixel value would be between `0-255` and not more than `MAX_VALUE` (255) for each colour. We will check that the pixel value of each colour does not exceed `colour` will get character by character values into buffer from all colour planes.


- Generating BLACK and WHITE of an Image:

Binary images are often produced by thresholding a grayscale or colour image, in order to separate an object in the image from the background. The colour of the object (usually white) is referred to as the *foreground colour*. The rest (usually black) is referred to as the *background colour*. However, depending on the image which is to be thresholder, this *polarity* might be inverted, in which case the object is displayed with 0 and the background is with a non-zero value.

to make the image black and white, we need to convert pixel values into 0(`BLACK`) and 255(`WHITE`) and store that values of pixels in buffer. For that, we replace all the pixel values above `THRESHOLD` (here 128) into 255 and below `THRESHOLD` into 0.

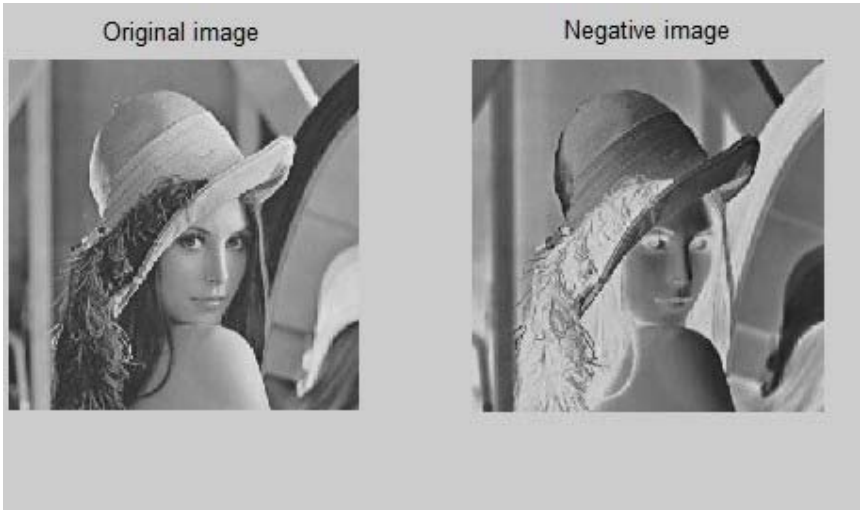

- Rotating an Image:

Rotating images by a given angle is a common image processing task. Although it seems little bit complicated, to rotate the image 180 degree, we need to swap the pixel values. For that, we replace all the columns to rows and vice versa. So, we will take out buffer and rotated values will be stored in out buffer. And thus, it will rotate our image.

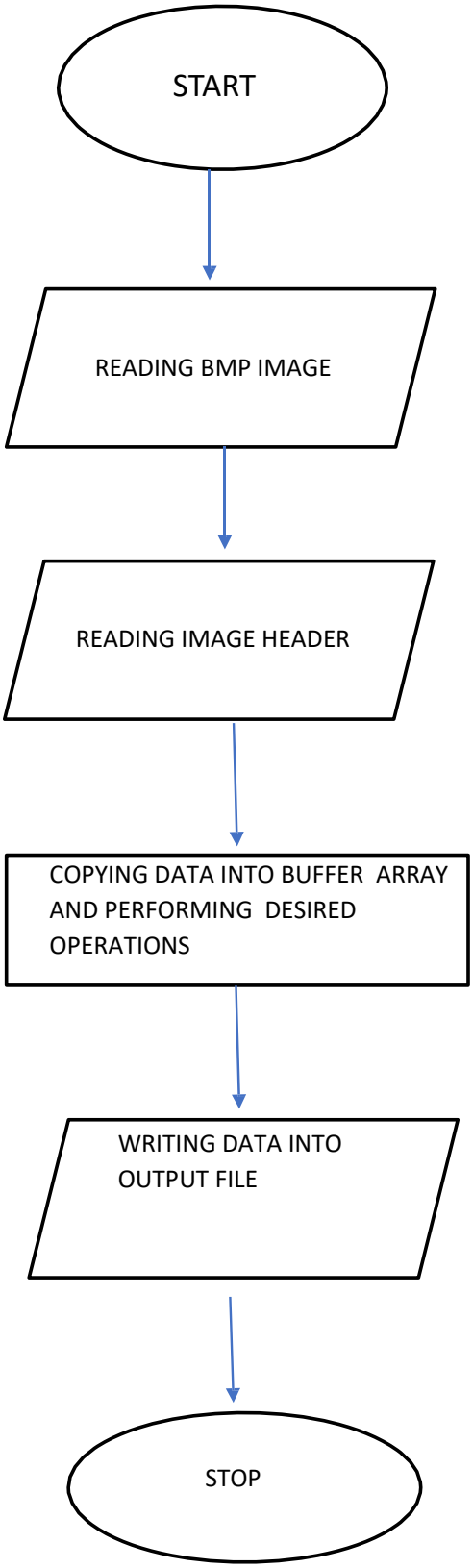
- Genrating Negative Of an Image:

Negative, photographic **image** that reproduces the bright portions of the photographed subject as dark and the dark parts as light areas. **Negatives** are usually formed on a transparent material, such as plastic or glass. A negative image is a total inversion, in which light areas appear dark and vice versa. A negative colour image is additionally colour-reversed,[2] with red areas appearing cyan, greens appearing magenta, and blues appearing yellow, and vice versa.

. To create the negative image, we need to have the complement of the colour which it already has. The input image in our case is a grey-scale image, hence we know that the pixel value would be between `0-255`. To find the complement, we just subtract the read image value from 255.



5) FLOWCHART


START
READING BMP IMAGE
READING IMAGE HEADER
COPYING DATA INTO BUFFER ARRAY AND PERFORMING DESIRED OPERATIONS
WRITING DATA INTO OUTPUT FILE
STOP

6) CODE SNIPPETS:

```
    #include <iostream>
#include <stdlib.h>
#include <string.h>
#include <conio.h>   ///to use _getche
#include <Windows.h> ///to  use  gotoxy and  setcolor
#include<stdio.h>
#define THRESHOLD 128
#define BLACK 0
#define WHITE 255
#define MAX_VALUE 255

using namespace std;

void gotoxy(short int, short int);
void setColor(int);

void gotoxy(short int x, short int y)
{
    COORD cur = {x, y};
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), cur);
}

bool loading()
{
    char ch;
    gotoxy(20, 23); /// Loading
    setColor(8);
    cout << "Loading...  ";
    gotoxy(5, 24);
    for (int i = 5; i <= 105; i++)
    {
        cout << "|";
        Sleep(15);
        gotoxy(37, 23);
        cout << i - 5 << " %";
        Sleep(15);
        gotoxy(i, 24);
    }
    setColor(12);
    gotoxy(30, 26);
    cout << "Press any key to continue ....";
    gotoxy(30, 27);
    cout<<"Press Esc to exit...";
```

```
    ch = _getch();
    if (ch == 27)
        return false;
    else
        return true;
}
void improcess()
{
    gotoxy(20, 23); /// Loading
    setColor(8);
    cout << "Applying Filter...  ";
    gotoxy(5, 24);
    for (int i = 5; i <= 105; i++)
    {
        cout << "|";
        Sleep(15);
        gotoxy(37, 23);
        cout << i - 5 << " %";
        Sleep(15);
        gotoxy(i, 24);
    }
    system("cls");
    gotoxy(30, 26);
    cout << "Image is saved in the designated folder ....";
}

void my_exit()
{
    system("cls");
    gotoxy(55, 13);
    setColor(11);
    cout << "Good Bye !!!";
    gotoxy(47, 17);
    _getch();
    exit(1);
}

void setColor(int c)
{ ///if (c < 15&& c> 0) font  color  else  background color
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    GetConsoleScreenBufferInfo(hConsole, &csbi);
    SetConsoleTextAttribute(hConsole, (csbi.wAttributes & 0xFFF0) | (WORD)c);
/// For ground colors take up the least significant byte
}

void name_and_rol_no()
{
```

```
    ///system("color fc");
    gotoxy(30, 15);
    setColor(11);
    cout << "PROJECT BY...";
    gotoxy(30, 18);
    setColor(14);
    char name[] = " ABRAR ZAHOOR AND AMAAN BILAL ";
    for (int i = 0; name[i] != '\0'; i++)
    {
        Sleep(80);
        cout << name[i];
    }
    setColor(14);
    gotoxy(30, 19);
    char rollno[] = " 2K19/IT/004   2K19/IT/011 ";
    for (int i = 0; rollno[i] != '\0'; i++)
    {
        Sleep(80);
        cout << rollno[i];
    }
}

int main()
{
    name_and_rol_no();
    bool flag =loading();
    char press;
    int n;
    setColor(11);
   hell:system("cls");
    if(flag)
    {
    hell1:system("cls");
    cout<<"\n\nPlease Select the Image filter :"<<endl;
    setColor(11);
    cout <<"\nPress 1 for RGB TO Grey" << endl;
    setColor(11);
    cout<<"\nPress 2 for RGB TO SPEIA"<<endl;
    cout<<"\nPress 3 for BLACK AND WHITE" << endl;
    cout<<"\nPress 4 to Rotate"<<endl;
    cout<<"\nPress 5 for Generating the Negative of the image"<<endl;
    cin>>n;

    switch (n)
    {
    case 1/* constant-expression */:
        /* code */
        {
```

```
            FILE *fIn = fopen("airplane.bmp", "rb");
            FILE *fOut = fopen("images/airplane_grey.bmp", "w+");
            int i, j, y;
            unsigned char byte[54];

            if (fIn == NULL) // Base case testing whether or not the input fil
e is present
            {
                cout << ("File does not exist.\n");
        }

        for (i = 0; i < 54; i++) //reading the 54 byte header via the input im
age from the function fIn
        {
            byte[i] = getc(fIn);
        }

        fwrite(byte, sizeof(unsigned char), 54, fOut); //writing back the head
er in the processed image

        // extracting image height, width and bitDepth from the input imageHea
der
        int height = *(int *)&byte[18];
        int width = *(int *)&byte[22];
        int bitDepth = *(int *)&byte[28];

        cout << "width: \n"
             << width << endl;
        cout << "height: \n"
             << height << endl;

        int size = height * width; //calculating the image size from the given
 dimensions

        unsigned char buffer[size][3]; //storing the image data extracted from
 the calculated size

        for (i = 0; i < size; i++) //Converting RGB colours to gray
        {
            y = 0;
            buffer[i][2] = getc(fIn); //blue
            buffer[i][1] = getc(fIn); //green
            buffer[i][0] = getc(fIn); //red

            y = (buffer[i][0] * 0.3) + (buffer[i][1] * 0.59) + (buffer[i][2] *
 0.11); //converting the RGB colors by multiplying with the respective values

            putc(y, fOut);
```

```
            putc(y, fOut);
            putc(y, fOut);
        }
        cout << endl<< endl;
        improcess();
         fclose(fOut);
         fclose(fIn);
        }
         break;
    case 2:
    {
        FILE *fIn = fopen("airplane.bmp", "rb");         //Input File name
        FILE *fOut = fopen("images/airplane_sepia.bmp", "w+"); //Output File n
ame


        int i, r, g, b;
        unsigned char byte[54];

        if (fIn == NULL) // check if the input file has not been opened succes
fully.
        {
            cout<<"File does not exist.\n";
        }

        for (i = 0; i < 54; i++) //read the 54 byte header from fIn
        {
            byte[i] = getc(fIn);
        }

        fwrite(byte, sizeof(unsigned char), 54, fOut); //write the header back

        // extract image height, width and bitDepth from imageHeader
        int height = *(int *)&byte[18];
        int width = *(int *)&byte[22];
        int bitDepth = *(int *)&byte[28];

        cout<<"width: \n"<<width;
        cout<<"height: \n"<< height;

        int size = height * width; //calculate image size

        unsigned char buffer[size][3]; //to store the image data

        for (i = 0; i < size; i++)
        {
            r = 0;
            g = 0;
            b = 0;
```

```
            buffer[i][2] = getc(fIn); //blue
            buffer[i][1] = getc(fIn); //green
            buffer[i][0] = getc(fIn); //red

            //conversion formula of rgb to sepia
            r = (buffer[i][0] * 0.393) + (buffer[i][1] * 0.769) + (buffer[i][2
] * 0.189);
            g = (buffer[i][0] * 0.349) + (buffer[i][1] * 0.686) + (buffer[i][2
] * 0.168);
            b = (buffer[i][0] * 0.272) + (buffer[i][1] * 0.534) + (buffer[i][2
] * 0.131);

            if (r > MAX_VALUE)
            { //if value exceeds
                r = MAX_VALUE;
            }
            if (g > MAX_VALUE)
            {
                g = MAX_VALUE;
            }
            if (b > MAX_VALUE)
            {
                b = MAX_VALUE;
            }
            putc(b, fOut);
            putc(g, fOut);
            putc(r, fOut);
        }

        fclose(fOut);
        fclose(fIn);
        cout<<endl<<endl;
        improcess();
    }
        break;
    case 3:{
        FILE *fIn = fopen("barbara.bmp", "rb"); //Input File name
        FILE *fOut = fopen("images/b_w.bmp", "wb");   //Output File name

        int i;
        unsigned char byte[54];         //to get the image header
        unsigned char colorTable[1024]; //to get the colortable

        if (fIn == NULL) // check if the input file has not been opened succes
fully.
        {
            cout<<"File does not exist.\n";
        }
```

```
        for (i = 0; i < 54; i++) //read the 54 byte header from fIn
        {
            byte[i] = getc(fIn);
        }

        fwrite(byte, sizeof(unsigned char), 54, fOut); //write the header back

        // extract image height, width and bitDepth from imageHeader
        int height = *(int *)&byte[18];
        int width = *(int *)&byte[22];
        int bitDepth = *(int *)&byte[28];

        cout<<"width: \n"<<width;
        cout<<"height: \n"<< height;
        int size = height * width; //calculate image size

        if (bitDepth <= 8) //if ColorTable present, extract it.
        {
            fread(colorTable, sizeof(unsigned char), 1024, fIn);
            fwrite(colorTable, sizeof(unsigned char), 1024, fOut);
        }

        unsigned char buffer[size]; //to store the image data

        fread(buffer, sizeof(unsigned char), size, fIn); //read image data

        for (i = 0; i < size; i++) //store 0(black) and 255(white) values to b
uffer
        {
            buffer[i] = (buffer[i] > THRESHOLD) ? WHITE : BLACK;
        }

        fwrite(buffer, sizeof(unsigned char), size, fOut); //write back to the
 output image

        fclose(fIn);
        fclose(fOut);
        cout << endl<< endl;
        improcess();

    }
        break;
    case 4:{
        FILE *fIn = fopen("barbara.bmp", "rb");       //Input File name
        FILE *fOut = fopen("images/barbara_rot.bmp", "wb"); //Output File name

        int i, j, choice;
```

```
        unsigned char byte[54], colorTable[1024];

        if (fIn == NULL) // check if the input file has not been opened succes
fully.
        {
            cout<<"File does not exist.\n";
        }

        for (i = 0; i < 54; i++) //read the 54 byte header from fIn
        {
            byte[i] = getc(fIn);
        }

        fwrite(byte, sizeof(unsigned char), 54, fOut); //write the header back

        // extract image height, width and bitDepth from imageHeader
        int height = *(int *)&byte[18];
        int width = *(int *)&byte[22];
        int bitDepth = *(int *)&byte[28];

        cout<<"width: \n"<<width;
        cout<<"height: \n"<< height;

        int size = height * width; //calculate image size

        if (bitDepth <= 8) //if ColorTable present, extract it.
        {
            fread(colorTable, sizeof(unsigned char), 1024, fIn);
            fwrite(colorTable, sizeof(unsigned char), 1024, fOut);
        }

        unsigned char buffer[width][height]; //to store the image data
        unsigned char out_buffer[width][height];

        fread(buffer, sizeof(unsigned char), size, fIn); //read the image data

        cout<<"Enter your choice :\n";
        cout<<"1. Rotate left\n";
        cout<<"2. Rotate right\n";
        cout<<"3. Rotate 180\n";

        cin>>choice;

        switch (choice) //to rotate image in 3 direction
        {
        case 1:
            for (i = 0; i < width; i++) //to rotate left
            {
```

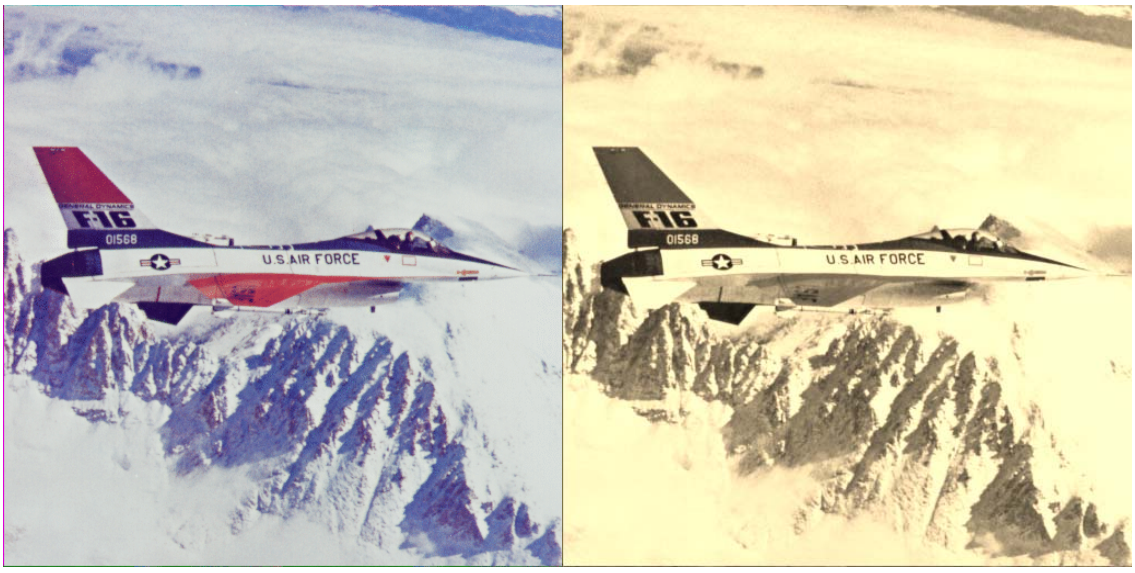
```
                for (j = 0; j < height; j++)
                {
                    out_buffer[j][height - 1 - i] = buffer[i][j];
                }
            }
            break;
        case 2:
            for (i = 0; i < width; i++) //to rotate right
            {
                for (j = 0; j < height; j++)
                {
                    out_buffer[j][i] = buffer[i][j];
                }
            }
            break;
        case 3:
            for (i = 0; i < width; i++) //to rotate 180 degree
            {
                for (j = 0; j < height; j++)
                {
                    out_buffer[width - i][j] = buffer[i][j];
                }
            }
            break;
        default:
            break;
        }

        fwrite(out_buffer, sizeof(unsigned char), size, fOut); //write back to
the output image

        fclose(fIn);
        fclose(fOut);

        cout << endl<< endl;
        improcess();
    }
        break;
    case 5 :{
        FILE *fp = fopen("barbara.bmp", "rb"); //read the file//

        unsigned char *imageData;       // to store the image information
        unsigned char *newimageData;    // to store the new image information,
i.e. the negative image
        unsigned char imageHeader[54];  // to get the image header
        unsigned char colorTable[1024]; // to get the colortable

        int i, j; // loop counter variables
```

```
        fread(imageHeader, sizeof(unsigned char), 54, fp); // read the 54-
byte from fp to imageHeader

        // extract image height and width from imageHeader
        int width = *(int *)&imageHeader[18];
        int height = *(int *)&imageHeader[22];
        int bitDepth = *(int *)&imageHeader[28];

        int imgDataSize = width * height; // calculate image size

        imageData = (unsigned char *)malloc(imgDataSize * sizeof(unsigned char
)); // allocate the block of memory as big as the image size
        newimageData = (unsigned char *)malloc(imgDataSize * sizeof(unsigned c
har));

        if (bitDepth <= 8)                                          // COLOR TABLE
 Present
            fread(colorTable, sizeof(unsigned char), 1024, fp); // read the 10
24-byte from fp to colorTable

        fread(imageData, sizeof(unsigned char), imgDataSize, fp);

        //Calculate the mean of the image
        for (i = 0; i < height; i++)
        {
            for (j = 0; j < width; j++)
            {
                newimageData[i * width + j] = 255 - imageData[i * width + j];
            }
        }

        FILE *fo = fopen("images/barbara_gray-negative.bmp", "wb");

        fwrite(imageHeader, sizeof(unsigned char), 54, fo); // write the heade
r back.

        if (bitDepth <= 8)                                           // COLOR TABL
E Present
            fwrite(colorTable, sizeof(unsigned char), 1024, fo); // write the
color table back

        fwrite(newimageData, sizeof(unsigned char), imgDataSize, fo); // write
 the values of the negative image.

        fclose(fo);
        fclose(fp);
        cout<<endl;
```

```
        improcess();
    }
        break;
    default:
        setColor(12);
        cout<<"NOT ABLE TO PROCESS THE IMAGE"<<endl;
        cout<<"ENTER A VALID CHOICE";
        _getch();
        goto hell1;
        break;

    }}
    else{
        my_exit();
    }
    setColor(10);
    cout<<"\n\nDo you want to apply any other Operation on the image(Press y/n
)"<<endl;
    cin>>press;
    if(press=='y'||press=='Y')
        goto hell;
        else
            my_exit();
    return 0;
 }
```

7) OUTPUT:

1. RGB TO SPEIA:



2. GREY TO BLACK and WHITE:

3. RGB TO GREY:

4. GREY TO NEGATIVE:

5. IMAGE ROTATION: (TOWARDS LEFT):

8) CONCLUSION:

Image processing has wide verity of applications beyond performing these mathematical operations on the image for various purposes leaving myriad options for the developer to choose one of the areas of his/her interest. Lots of research findings are published but lots of research areas are still untouched. Moreover, with the fast computers and signal processors available in the 2000s, digital image processing has become the most common form of image processing and generally, is used because it is not only the most versatile method, but also the cheapest.