Name: Deepankar Sharma course: BCA-6th roll no: 2092014
Subject: Computer Graphics

Index

NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

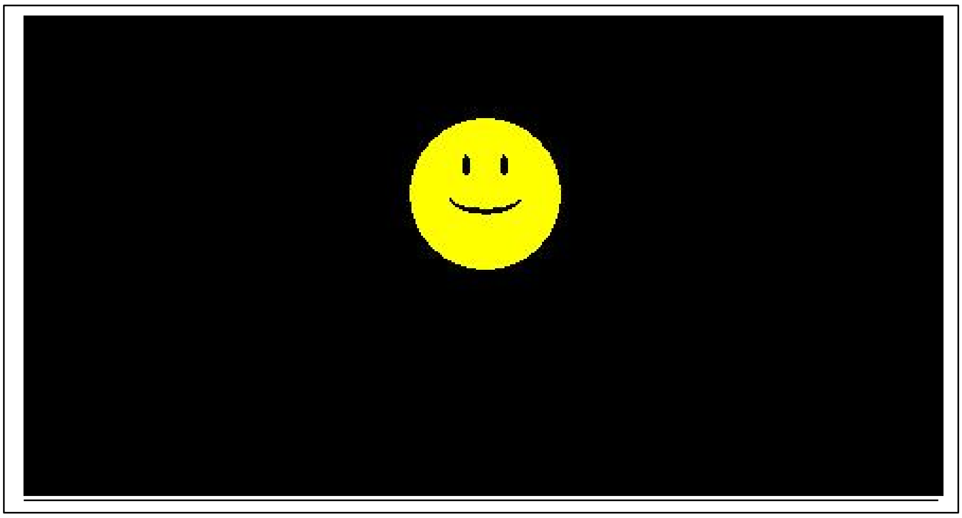
PRACTICLE-1

OBJECTIVE- DRAW A SMILEY FACE THOUGH GRAPHICS

SYNTAX :-

```
#include <graphics.h>

int main()
{
    int gr = DETECT, gm;
    initgraph(&gr, &gm, "C:\\Turboc3\\BGI");
    setcolor(YELLOW);
    circle(300, 100, 40);
    setfillstyle(SOLID_FILL, YELLOW);
    floodfill(300, 100, YELLOW);
    setcolor(BLACK);
    setfillstyle(SOLID_FILL, BLACK);
    fillellipse(310, 85, 2, 6);
    fillellipse(290, 85, 2, 6);
    ellipse(300, 100, 205, 335, 20, 9);
    ellipse(300, 100, 205, 335, 20, 10);
    ellipse(300, 100, 205, 335, 20, 11);
      getch();
    closegraph();

    return 0;
}}
```

OUTPUT:



NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

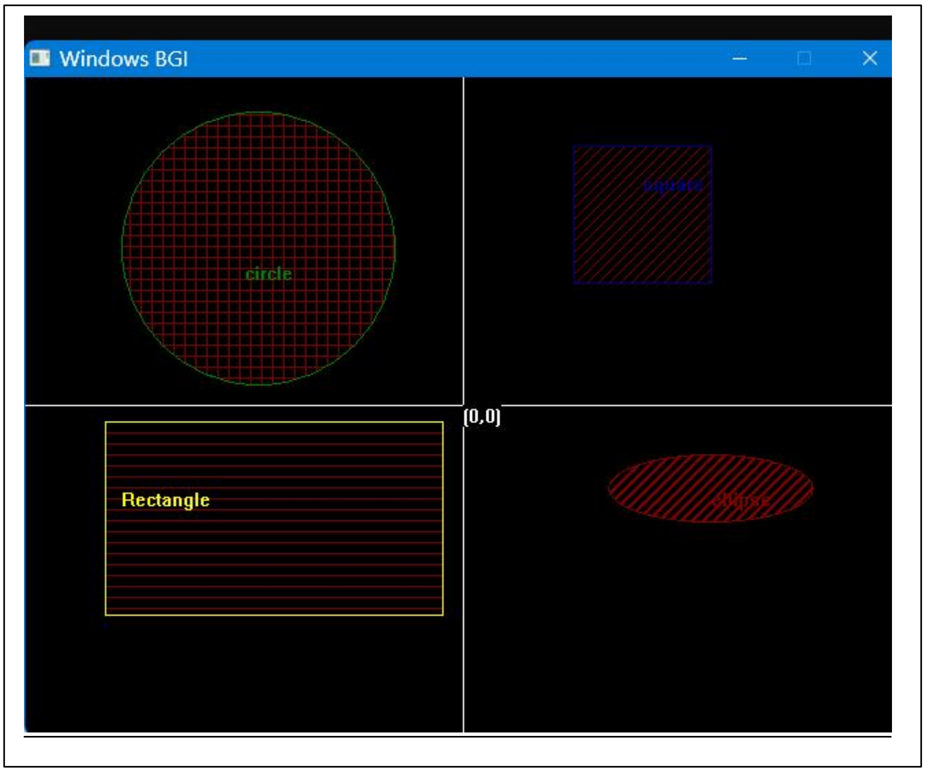
PRACTICLE-2

OBJECTIVE- To divide your screen into four region, draw circle, rectangle, ellipse ,square.

SYNTAX :-

```
#include<conio.h>
#include<graphics.h>
#include<stdio.h>
int main()
{
int gdriver = DETECT, gmode;
int xmax,ymax;
initgraph(&gdriver, &gmode,"c:\\turboc3\\bgi");
xmax = getmaxx();
ymax = getmaxy();
line(xmax/2,0,xmax/2,ymax);
line(0,ymax/2,xmax,ymax/2);
outtextxy (xmax/2,ymax/2,"(0,0)");

setcolor(GREEN);
setfillstyle(HATCH_FILL,RED);
circle(170,125,100);
outtextxy (160,135,"circle");
floodfill(170,125,GREEN);


setcolor(YELLOW);
setfillstyle(2,RED);
rectangle(58,251,304,392);
outtextxy (70,300,"Rectangle");
floodfill(70,351,YELLOW);


setcolor(BLUE);
setfillstyle(3,RED);
rectangle(400,50,500,150);
outtextxy (450,70,"square");
floodfill(450,80,BLUE);

setcolor(RED);
setfillstyle(4,RED);
ellipse(500,300,0,360,75,25);
outtextxy (500,300,"ellipse");
floodfill(500,300,RED);

getch();
closegraph();
return 0;
}
```

OUTPUT:



NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

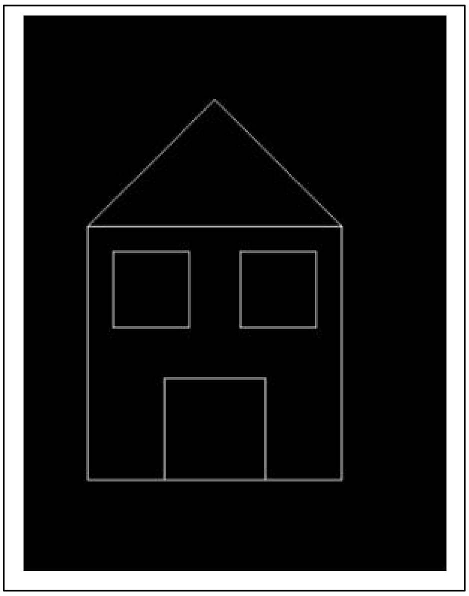
PRACTICLE- 3

OBJECTIVE- DRAW A HOUSE THOUGH GRAPHICS

 SYNTAX :-

```
#include <graphics.h>

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   rectangle(100, 200, 300, 400);
   line(100, 200, 200, 100);
   line(200, 100, 300, 200);
   rectangle(120, 220, 180, 280);
   rectangle(220, 220, 280, 280);
   rectangle(160, 320, 240, 400);

   getch();
   closegraph();
   return 0;
}
```

OUTPUT:



NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

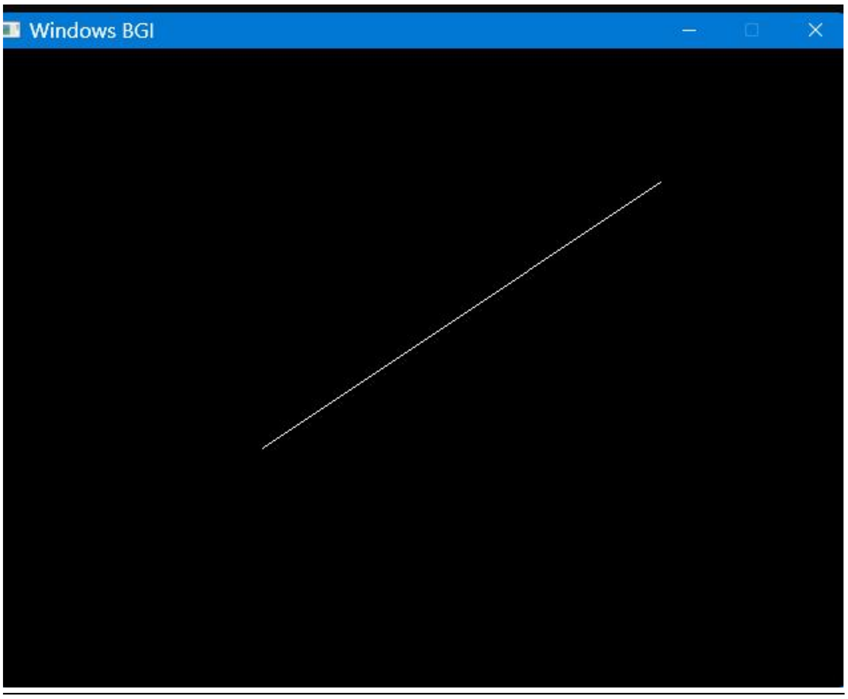
PRACTICLE-4

OBJECTIVE- TO IMPLEMENT THE DDA LINE GENERATION ALGORITHM THOUGH GRAPHICS

SYNTAX :-

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
int main()
{
    int gd = DETECT ,gm, i;
    float x, y,dx,dy,steps;
    int x0, x1, y0, y1;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    x0 = 200 , y0 = 300, x1 = 500, y1 = 100;
    dx = (float)(x1 - x0);
    dy = (float)(y1 - y0);
    if(dx>=dy)
          {
       steps = dx;
    }
    else
          {
       steps = dy;
    }
    dx = dx/steps;
    dy = dy/steps;
    x = x0;
    y = y0;
    i = 1;
    while(i<= steps)
    {
       putpixel(x, y, WHITE);
       x += dx;
       y += dy;
       i=i+1;
    }
    getch();
    closegraph();
}
```

OUTPUT:



NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

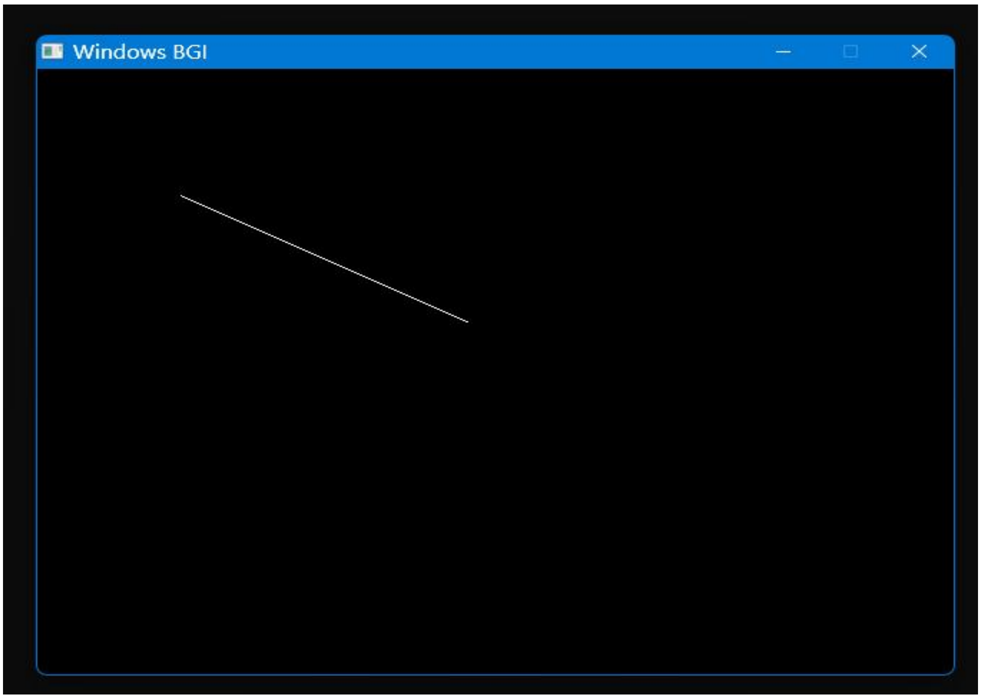
PRACTICLE-5

OBJECTIVE- TO IMPLEMENT THE Bresenham's Line Algorithm THOUGH GRAPHICS

SYNTAX :-

```
#include <iostream>
#include <graphics.h>
void bresenham(int x1, int y1, int x2, int y2) {
    int dx = x2 - x1;
    int dy = y2 - y1;
    int p = 2 * dy - dx;
    int twoDy = 2 * dy;
    int twoDyMinusDx = 2 * (dy - dx);
    int x = x1;
    int y = y1;

    if (x1 > x2) {
        x = x2;
        y = y2;
        x2 = x1;
    } else {
        x = x1;
        y = y1;
    }

    putpixel(x, y, WHITE);

    while (x < x2) {
        x++;
        if (p < 0) {
            p += twoDy;
        } else {
            y++;
            p += twoDyMinusDx;
        }
        putpixel(x, y, BLUE);
    }
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    bresenham(100, 100, 300, 200);
    getch();
    closegraph();
}
```

OUTPUT:



NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

PRACTICLE-6

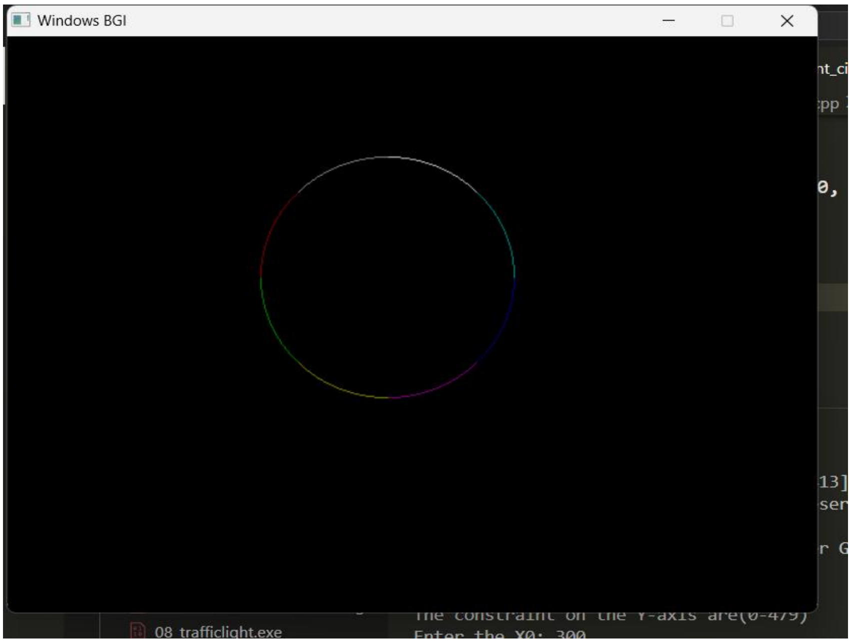
OBJECTIVE- To implement Mid Point Circle drawing Algorithm through graphics.

SYNTAX:-

```
#include<graphics.h>
#include<iostream>
using namespace std;

// Midpoint Circle drawing Algorithm
void drawMidPointCircle(int x0, int y0, int radius)
{
    int x = radius, y = 0;
    int decisionParam = 1 - radius;

    while (y <= x)
    {
        putpixel(x0 + x, y0 + y, 1);
        putpixel(x0 - x, y0 + y, 2);
        putpixel(x0 + x, y0 - y, 3);
        putpixel(x0 - x, y0 - y, 4);
        putpixel(x0 + y, y0 + x, 5);
        putpixel(x0 - y, y0 + x, 6);
        putpixel(x0 + y, y0 - x, 7);
        putpixel(x0 - y, y0 - x, 8);

        y++;

        if (decisionParam <= 0)
            decisionParam += 2 * y + 1;
        else
        {
            x--;
            decisionParam += 2 * (y - x) + 1;
        }
    }
}

int main()
{
    int gDrive = DETECT;

    int gMode;
    initgraph(&gDrive, &gMode, NULL);

    int X0 = 0, Y0 = 0, radius=0 ;

    printf("The constraint on the X-axis are(0-%d)\n", getmaxx());
    printf("The constraint on the Y-axis are(0-%d)\n", getmaxy());

    cout<<("Enter the X0: ");
    scanf("%d", &X0);
    cout<<("Enter the Y0: ");
    scanf("%d", &Y0);
    cout<<("Enter the radius: ");
    scanf("%d", &radius);


    // Function call
    // DDA(X0, Y0, X1, Y1);
    drawMidPointCircle(X0, Y0, radius);
    // DDA(2, 2, 14, 16);
    getch();
    closegraph();
    return 0;
}
```

OUTPUT:-

```
The constraint on the X-axis are(0-639)
The constraint on the Y-axis are(0-479)
Enter the X0: 300
Enter the Y0: 200
Enter the radius: 100
```



NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

PRACTICLE-7

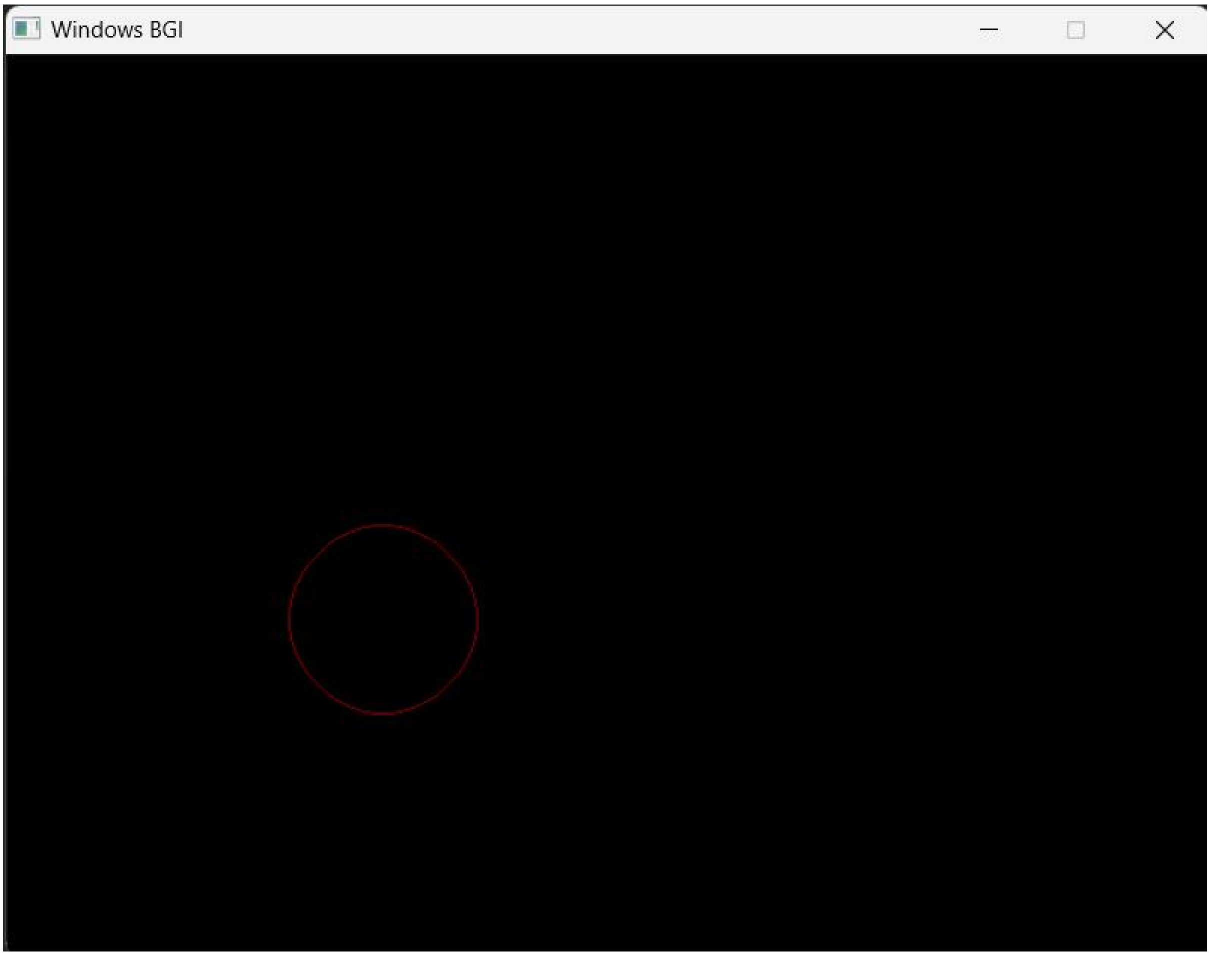
OBJECTIVE- To implement Brensanham’s Circle drawing Algorithm through graphics.

SYNTAX:-

```
#include <graphics.h>
#include <iostream>
using namespace std;

// Midpoint Circle drawing Algorithm
void drawMidPointCircle(int x0, int y0, int radius)
{
    int x = radius, y = 0;
    int decisionParam = 1 - radius;

    while (y <= x)
    {
        putpixel(x0 + x, y0 + y, 1);
        putpixel(x0 - x, y0 + y, 2);
        putpixel(x0 + x, y0 - y, 3);
        putpixel(x0 - x, y0 - y, 4);
        putpixel(x0 + y, y0 + x, 5);
        putpixel(x0 - y, y0 + x, 6);
        putpixel(x0 + y, y0 - x, 7);
        putpixel(x0 - y, y0 - x, 8);

        y++;

        if (decisionParam <= 0)
            decisionParam += 2 * y + 1;
        else
        {
            x--;
            decisionParam += 2 * (y - x) + 1;
        }
    }
}

// Brensanham Circle drawing Algorithm

void drawBrensanhamCircle(int x0, int y0, int radius)
{
    int x = 0, y = radius;
    int decisionParam = 3 - 2 * radius;
    while (x <= y)
    {
        putpixel(x0 + x, y0 + y, RED);
        putpixel(x0 + y, y0 + x, RED);
        putpixel(x0 - y, y0 + x, RED);
        putpixel(x0 - x, y0 + y, RED);
        putpixel(x0 - x, y0 - y, RED);
        putpixel(x0 - y, y0 - x, RED);
        putpixel(x0 + y, y0 - x, RED);
        putpixel(x0 + x, y0 - y, RED);

        if (decisionParam <= 0)
        {
            x++;
            decisionParam += 4 * x + 6;
        }
        else
        {
            x++;
            y--;
            decisionParam += 4 * (x - y) + 10;
        }
    }
}

int main()
{
    int gDrive = DETECT;

    int gMode;

    initgraph(&gDrive, &gMode, NULL);

    int X0 = 0, Y0 = 0, radius = 0;

    printf("The constraint on the X-axis are(0-%d)\n", getmaxx());
    printf("The constraint on the Y-axis are(0-%d)\n", getmaxy());

    cout << ("Enter the X0: ");
    scanf("%d", &X0);
    cout << ("Enter the Y0: ");
    scanf("%d", &Y0);
    cout << ("Enter the radius: ");
    scanf("%d", &radius);

    // Function call
    // DDA(X0, Y0, X1, Y1);
    drawBrensanhamCircle(X0, Y0, radius);
    // DDA(2, 2, 14, 16);
    getch();
    closegraph();
    return 0;
}
```

OUTPUT:-

```
C:\Deepankar\06_semester\TBC 601 Computer
Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer
Graphics\PracticalsVScode\Home\build\07_Bresenham_circleDrawing.exe"
The constraint on the X-axis are(0-639)
The constraint on the Y-axis are(0-479)
Enter the X0: 200
Enter the Y0: 300
Enter the radius: 50
```



NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

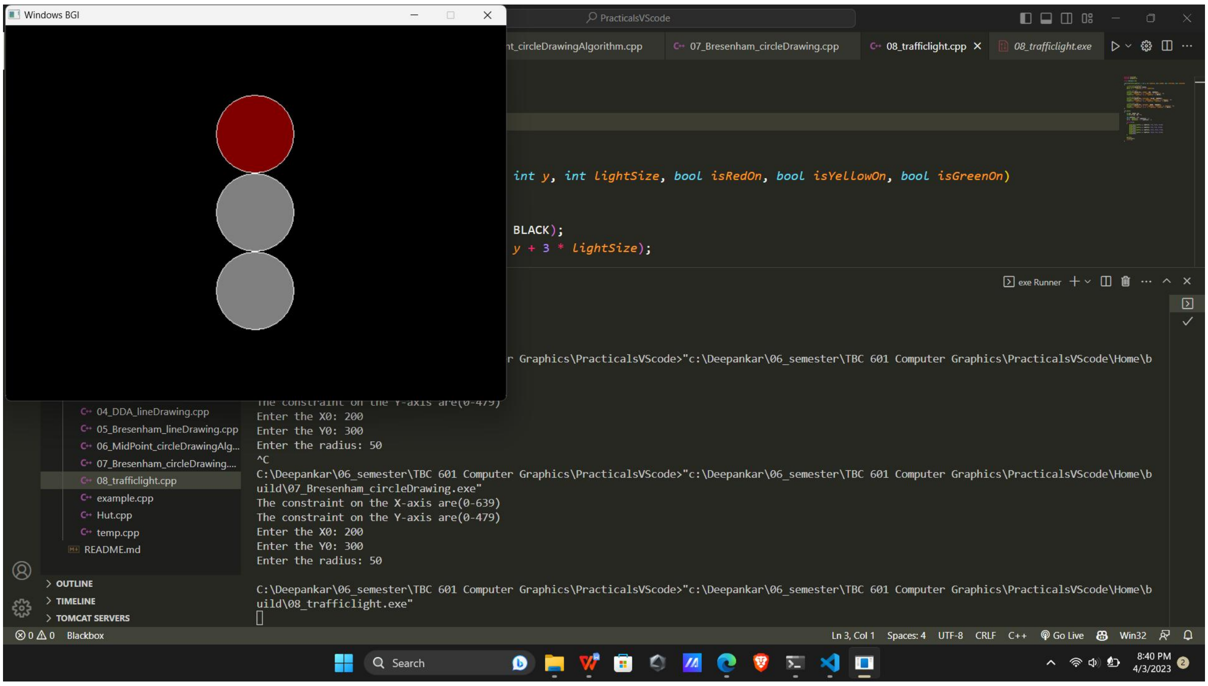
PRACTICLE-8

OBJECTIVE- To implement Blinking Traffic Light through graphics.

SYNTAX:-

```
#include <iostream>
#include <graphics.h>

using namespace std;

void drawTrafficLight(int x, int y, int lightSize, bool isRedOn,
bool isYellowOn, bool isGreenOn)
{
    // Draw black background
    setfillstyle(SOLID_FILL, BLACK);
    bar(x, y, x + lightSize, y + 3 * lightSize);

    // Draw red light
    setfillstyle(SOLID_FILL, isRedOn ? RED : DARKGRAY);
    circle(x + lightSize / 2, y + lightSize / 2, lightSize / 2);
    floodfill(x + lightSize / 2, y + lightSize / 2, WHITE);

    // Draw yellow light
    setfillstyle(SOLID_FILL, isYellowOn ? YELLOW : DARKGRAY);
    circle(x + lightSize / 2, y + lightSize + lightSize / 2,
lightSize / 2);
    floodfill(x + lightSize / 2, y + lightSize + lightSize / 2,
WHITE);

    // Draw green light
    setfillstyle(SOLID_FILL, isGreenOn ? GREEN : DARKGRAY);
    circle(x + lightSize / 2, y + 2 * lightSize + lightSize / 2,
lightSize / 2);
    floodfill(x + lightSize / 2, y + 2 * lightSize + lightSize / 2,
WHITE);
}

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int lightSize = 100;
    int x = (getmaxx() - lightSize) / 2;
    int y = (getmaxy() - 3 * lightSize) / 2;

    while (true)
    {
        drawTrafficLight(x, y, lightSize, true, false, false);
        delay(400);
        drawTrafficLight(x, y, lightSize, true, true, false);
        delay(400);
        drawTrafficLight(x, y, lightSize, false, false, true);
        delay(400);
        drawTrafficLight(x, y, lightSize, false, true, false);
        delay(400);
    }
    getch();
    closegraph();
    return 0;
}
```

OUTPUT:-



```
int y, int lightSize, bool isRedOn, bool isYellowOn, bool isGreenOn)

BLACK);
y + 3 * lightSize);
```

```
r Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode\Home\b

The constraint on the Y-axis are(0-479)
Enter the X0: 200
Enter the Y0: 300
Enter the radius: 50
^C
C:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode\Home\b
uild\07_Bresenham_circleDrawing.exe"
The constraint on the X-axis are(0-639)
The constraint on the Y-axis are(0-479)
Enter the X0: 200
Enter the Y0: 300
Enter the radius: 50

C:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode\Home\b
uild\08_trafficlight.exe"
```

NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

PRACTICLE-9

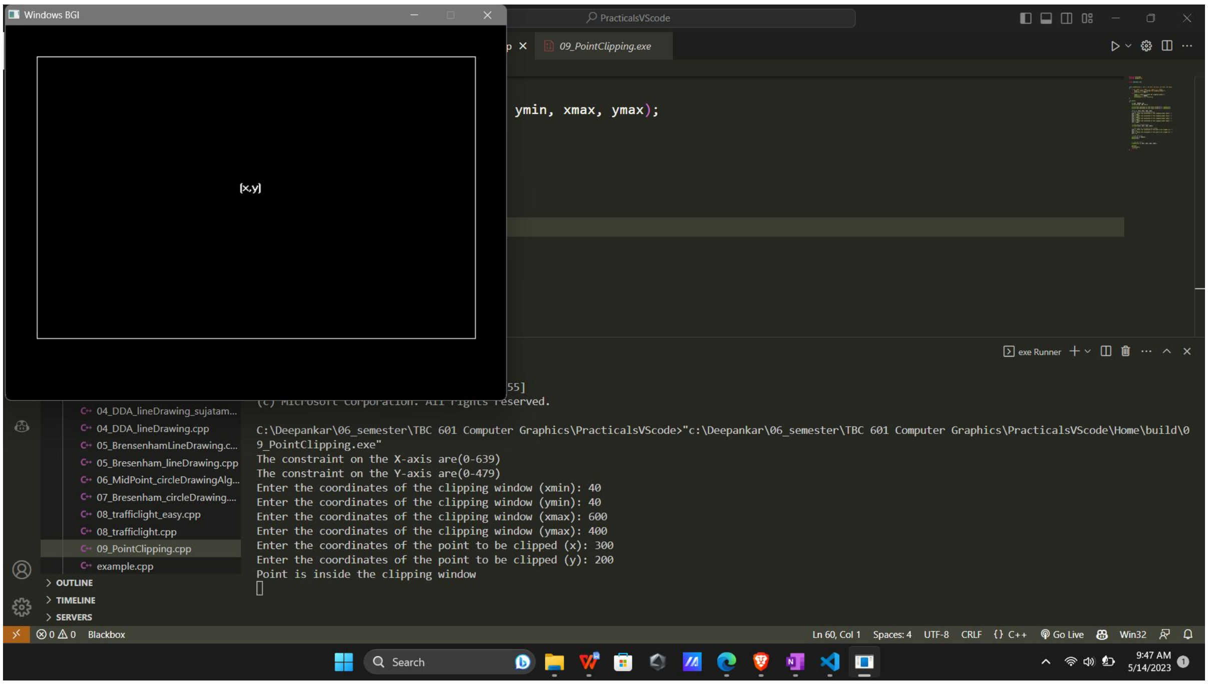
OBJECTIVE- To implement Point Clipping through graphics.

SYNTAX:-

```
#include <iostream>
#include <graphics.h>


using namespace std;



void clipPoint(int x, int y, int xmin, int ymin, int xmax, int ymax)
{
    if (x < xmin || x > xmax || y < ymin || y > ymax)
        {cout << "Point is outside the clipping window\n";
        putpixel(x, y, RED);}
    else
        {cout << "Point is inside the clipping window\n";
        putpixel(x, y, GREEN);
        outtextxy(x-1, y-1, "(x,y)");}
}

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    printf("The constraint on the X-axis are(0-%d)\n", getmaxx());
    printf("The constraint on the Y-axis are(0-%d)\n", getmaxy());

    int x, y, xmin, ymin, xmax, ymax;
    // take input for the clipping window
    cout << "Enter the coordinates of the clipping window (xmin): ";
    cin >> xmin;
    cout << "Enter the coordinates of the clipping window (ymin): ";
    cin >> ymin;
    cout << "Enter the coordinates of the clipping window (xmax): ";
    cin >> xmax ;
    cout << "Enter the coordinates of the clipping window (ymax): ";
    cin >> ymax;

    // draw the clipping window
    rectangle(xmin, ymin, xmax, ymax);
    // take input for the point to be clipped
    cout << "Enter the coordinates of the point to be clipped (x): ";
    cin >> x;
    cout << "Enter the coordinates of the point to be clipped (y): ";
    cin >> y;

    // draw the point
    putpixel(x, y, WHITE);
    delay(1000);
    // clip the point
    clipPoint(x, y, xmin, ymin, xmax, ymax);

    getch();
    closegraph();
    return 0;
}
```

OUTPUT:-

```
C:\Deepankar\06_semester\TBC 601 Computer
Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer
Graphics\PracticalsVScode\Home\build\09_PointClipping.exe"
The constraint on the X-axis are(0-639)
The constraint on the Y-axis are(0-479)
Enter the coordinates of the clipping window (xmin): 40
Enter the coordinates of the clipping window (ymin): 40
Enter the coordinates of the clipping window (xmax): 600
Enter the coordinates of the clipping window (ymax): 400
Enter the coordinates of the point to be clipped (x): 300
Enter the coordinates of the point to be clipped (y): 200
Point is inside the clipping window
```



```
ymin, xmax, ymax);
```

```
55]
(c) Microsoft Corporation. All rights reserved.

C:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode\Home\build\0
9_PointClipping.exe"
The constraint on the X-axis are(0-639)
The constraint on the Y-axis are(0-479)
Enter the coordinates of the clipping window (xmin): 40
Enter the coordinates of the clipping window (ymin): 40
Enter the coordinates of the clipping window (xmax): 600
Enter the coordinates of the clipping window (ymax): 400
Enter the coordinates of the point to be clipped (x): 300
Enter the coordinates of the point to be clipped (y): 200
Point is inside the clipping window
```

NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

PRACTICLE-10

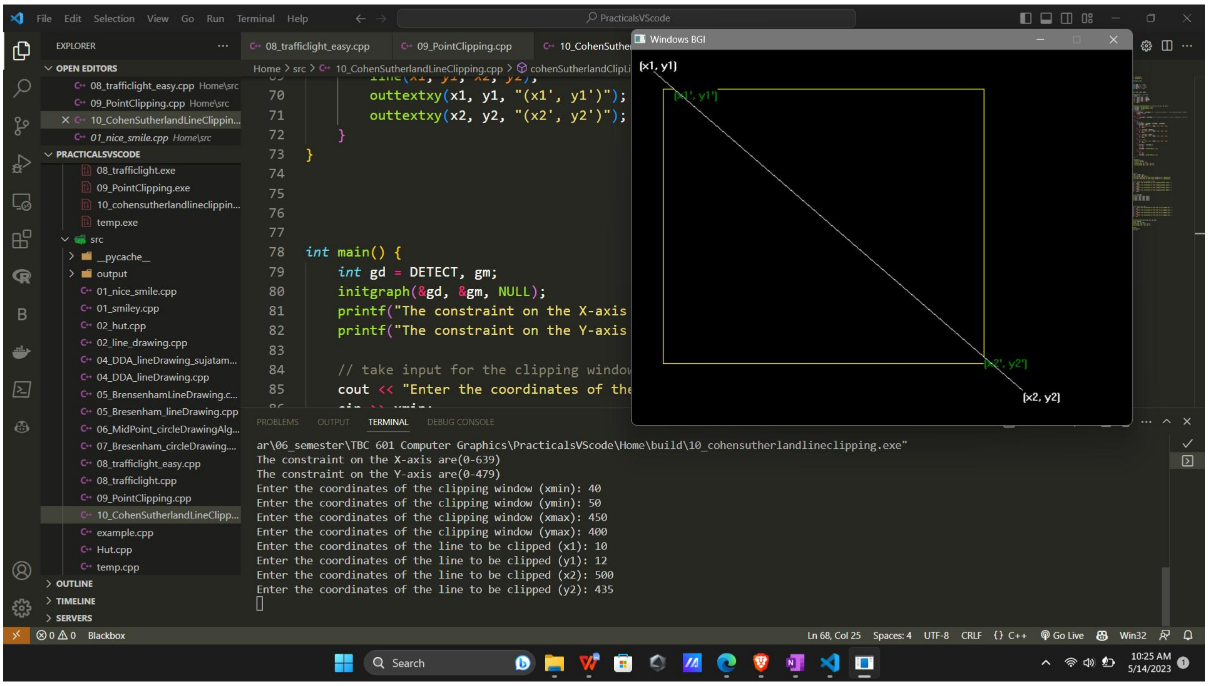
OBJECTIVE- To implement Cohen Sutherland Line Clipping Algorithm through graphics.

SYNTAX:-

```
#include <iostream>
#include <graphics.h>

using namespace std;


const int LEFT= 1; // 0001
const int RIGHT= 2; // 0010
const int BOTTOM= 4; // 0100
const int TOP= 8; // 1000


int xmin, xmax, ymin, ymax;

int getOutcode(int x, int y) {
    int code = 0;
    if (x < xmin) code |= LEFT;
    if (x > xmax) code |= RIGHT;
    if (y < ymin) code |= BOTTOM;
    if (y > ymax) code |= TOP;
    return code;
}

void cohenSutherlandClipLine(int x1, int y1, int x2, int y2) {
    int outcode1 = getOutcode(x1, y1);
    int outcode2 = getOutcode(x2, y2);
    bool accept = false;
    while (true) {
        if (!(outcode1 | outcode2)) { // trivially accepted -> line
clipping window k andar h
            accept = true;
            break;
        }
        else if (outcode1 & outcode2) { // trivially rejected -> line
is completely invisible
            break;
        }
        else {
            int x, y;
            int outcode = outcode1 ? outcode1 : outcode2;
            if (outcode & TOP) {
                x = x1 + (x2 - x1) * (ymax - y1) / (y2 - y1);
                y = ymax;
            }
            else if (outcode & BOTTOM) {
                x = x1 + (x2 - x1) * (ymin - y1) / (y2 - y1);
                y = ymin;
            }
            else if (outcode & RIGHT) {
                y = y1 + (y2 - y1) * (xmax - x1) / (x2 - x1);
                x = xmax;
            }
            else { // LEFT
                y = y1 + (y2 - y1) * (xmin - x1) / (x2 - x1);
                x = xmin;
            }
            if (outcode == outcode1) {
                x1 = x;
                y1 = y;
                outcode1 = getOutcode(x1, y1);
            }
            else {
                x2 = x;
                y2 = y;
                outcode2 = getOutcode(x2, y2);
            }
        }
    }
    if (accept) {
        setcolor(GREEN);
        line(x1, y1, x2, y2);
        outtextxy(x1, y1, "(x1', y1')");
        outtextxy(x2, y2, "(x2', y2')");
    }
}



int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    printf("The constraint on the X-axis are(0-%d)\n", getmaxx());
    printf("The constraint on the Y-axis are(0-%d)\n", getmaxy());

    // take input for the clipping window
    cout << "Enter the coordinates of the clipping window (xmin): ";
    cin >> xmin;
    cout << "Enter the coordinates of the clipping window (ymin): ";
    cin >> ymin;
    cout << "Enter the coordinates of the clipping window (xmax): ";
    cin >> xmax;
    cout << "Enter the coordinates of the clipping window (ymax): ";
    cin >> ymax;


    setcolor(YELLOW);
    line(xmin, ymin, xmax, ymin);
    line(xmax, ymin, xmax, ymax);
    line(xmax, ymax, xmin, ymax);
    line(xmin, ymax, xmin, ymin);




    int x1 , y1 , x2 , y2 ;
    cout << "Enter the coordinates of the line to be clipped (x1): ";
    cin >> x1;
    cout << "Enter the coordinates of the line to be clipped (y1): ";
    cin >> y1;
    cout << "Enter the coordinates of the line to be clipped (x2): ";
    cin >> x2;
    cout << "Enter the coordinates of the line to be clipped (y2): ";
    cin >> y2;


    cohenSutherlandClipLine(x1, y1, x2, y2);
    setcolor(WHITE);
    line(x1, y1, x2, y2);
    outtextxy(x1, y1, "(x1, y1)");
    outtextxy(x2, y2, "(x2, y2)");

    getch();
    closegraph();
    return 0;
}
```

OUTPUT:-

```
C:\Deepankar\06_semester\TBC 601 Computer
Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer
Graphics\PracticalsVScode\Home\build\10_cohensutherlandlineclipping.
exe"
The constraint on the X-axis are(0-639)
The constraint on the Y-axis are(0-479)
Enter the coordinates of the clipping window (xmin): 40
Enter the coordinates of the clipping window (ymin): 50
Enter the coordinates of the clipping window (xmax): 450
Enter the coordinates of the clipping window (ymax): 400
Enter the coordinates of the line to be clipped (x1): 10
Enter the coordinates of the line to be clipped (y1): 12
Enter the coordinates of the line to be clipped (x2): 500
Enter the coordinates of the line to be clipped (y2): 435
```



```
        line(x1, y1, x2, y2);
        outtextxy(x1, y1, "(x1', y1')");
        outtextxy(x2, y2, "(x2', y2')");
    }
}



int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    printf("The constraint on the X-axis
    printf("The constraint on the Y-axis

    // take input for the clipping window
    cout << "Enter the coordinates of the
```

```
ar\06_semester\TBC 601 Computer Graphics\PracticalsVScode\Home\build\10_cohensutherlandlineclipping.exe"
The constraint on the X-axis are(0-639)
The constraint on the Y-axis are(0-479)
Enter the coordinates of the clipping window (xmin): 40
Enter the coordinates of the clipping window (ymin): 50
Enter the coordinates of the clipping window (xmax): 450
Enter the coordinates of the clipping window (ymax): 400
Enter the coordinates of the line to be clipped (x1): 10
Enter the coordinates of the line to be clipped (y1): 12
Enter the coordinates of the line to be clipped (x2): 500
Enter the coordinates of the line to be clipped (y2): 435
```

NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

PRACTICLE-11

OBJECTIVE- To implement Liang Barsky Line Clipping Algorithm through graphics.

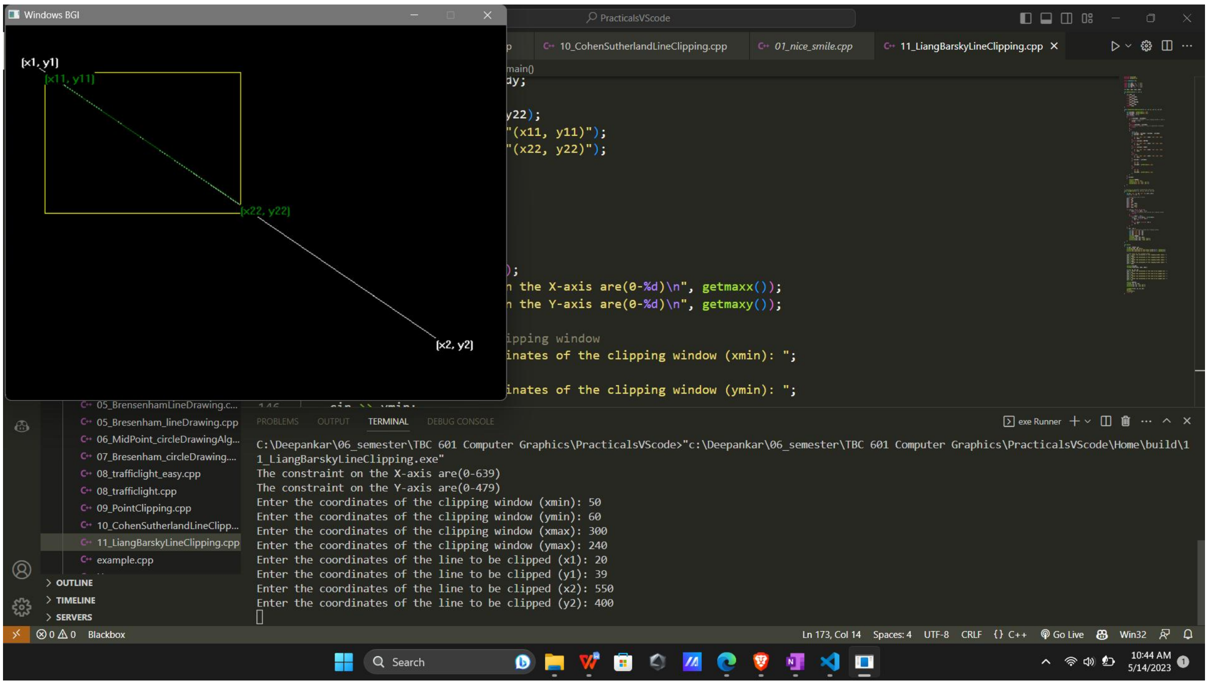
SYNTAX:-

```
#include <iostream>
#include <graphics.h>

using namespace std;

const int LEFT = 1;   // 0001
const int RIGHT = 2;  // 0010
const int BOTTOM = 4; // 0100
const int TOP = 8;    // 1000

int xmin, xmax, ymin, ymax;

int getOutcode(int x, int y)
{
    int code = 0;
    if (x < xmin)
```

```
        code |= LEFT;
    if (x > xmax)
        code |= RIGHT;
    if (y < ymin)
        code |= BOTTOM;
    if (y > ymax)
        code |= TOP;
    return code;
}

void cohenSutherlandClipLine(int x1, int y1, int x2, int y2)
{
    int outcode1 = getOutcode(x1, y1);
    int outcode2 = getOutcode(x2, y2);
    bool accept = false;
    while (true)
    {
        if (!(outcode1 | outcode2))
        { // trivially accepted -> line clipping window k andar h
            accept = true;
            break;
        }
        else if (outcode1 & outcode2)
        { // trivially rejected -> line is completely invisible
            break;
        }
        else
        {
            int x, y;
            int outcode = outcode1 ? outcode1 : outcode2;
            if (outcode & TOP)
            {
                x = x1 + (x2 - x1) * (ymax - y1) / (y2 - y1);
                y = ymax;
            }
            else if (outcode & BOTTOM)
            {
                x = x1 + (x2 - x1) * (ymin - y1) / (y2 - y1);
                y = ymin;
            }
            else if (outcode & RIGHT)
            {
                y = y1 + (y2 - y1) * (xmax - x1) / (x2 - x1);
                x = xmax;
            }
            else
            { // LEFT
                y = y1 + (y2 - y1) * (xmin - x1) / (x2 - x1);
                x = xmin;
            }
            if (outcode == outcode1)
            {
                x1 = x;
                y1 = y;
                outcode1 = getOutcode(x1, y1);
            }
            else
            {
                x2 = x;
                y2 = y;
                outcode2 = getOutcode(x2, y2);
            }
        }
    }
    if (accept)
    {
        setcolor(GREEN);
        line(x1, y1, x2, y2);
        outtextxy(x1, y1, "(x1', y1')");
        outtextxy(x2, y2, "(x2', y2')");
    }
}


// Function to implement Liang-Barsky algorithm
void LiangBarsky(int x1, int y1, int x2, int y2)
{
    int dx = x2 - x1, dy = y2 - y1, p[4], q[4];
    float t1 = 0, t2 = 1;

    // Calculating p and q values
    p[0] = -dx;
    p[1] = dx;
    p[2] = -dy;
    p[3] = dy;
    q[0] = x1 - xmin;
    q[1] = xmax - x1;
    q[2] = y1 - ymin;
    q[3] = ymax - y1;

    for (int i = 0; i < 4; i++) {
        if (p[i] == 0 && q[i] < 0) {
            // Line is parallel and outside the clipping window
            return;
        }
        else if (p[i] != 0) {
            float t = (float)q[i] / (float)p[i];
            if (p[i] < 0 && t > t1) {
                t1 = t;
            }
            else if (p[i] > 0 && t < t2) {
                t2 = t;
            }
        }
    }
    if (t1 < t2) {
        // Line is partially inside the clipping window
        int x11 = x1 + t1 * dx;
        int y11 = y1 + t1 * dy;
        int x22 = x1 + t2 * dx;
        int y22 = y1 + t2 * dy;
        setcolor(GREEN);
        line(x11, y11, x22, y22);
        outtextxy(x11, y11, "(x11, y11)");
        outtextxy(x22, y22, "(x22, y22)");
    }
}

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    printf("The constraint on the X-axis are(0-%d)\n", getmaxx());
    printf("The constraint on the Y-axis are(0-%d)\n", getmaxy());

    // take input for the clipping window
    cout << "Enter the coordinates of the clipping window (xmin): ";
    cin >> xmin;
    cout << "Enter the coordinates of the clipping window (ymin): ";
    cin >> ymin;
    cout << "Enter the coordinates of the clipping window (xmax): ";
    cin >> xmax;
    cout << "Enter the coordinates of the clipping window (ymax): ";
    cin >> ymax;

    setcolor(YELLOW);
    rectangle(xmin,ymin, xmax, ymax);

    int x1, y1, x2, y2;
    cout << "Enter the coordinates of the line to be clipped (x1): ";
    cin >> x1;
    cout << "Enter the coordinates of the line to be clipped (y1): ";
    cin >> y1;
    cout << "Enter the coordinates of the line to be clipped (x2): ";
    cin >> x2;
    cout << "Enter the coordinates of the line to be clipped (y2): ";
    cin >> y2;

    setcolor(WHITE);
    line(x1, y1, x2, y2);
    outtextxy(x1, y1, "(x1, y1)");
    outtextxy(x2, y2, "(x2, y2)");

    LiangBarsky(x1, y1, x2, y2);
    getch();
    closegraph();
    return 0;
}
```

OUTPUT:-

```
C:\Deepankar\06_semester\TBC 601 Computer
Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer
Graphics\PracticalsVScode\Home\build\11_LiangBarskyLineClipping.exe"
The constraint on the X-axis are(0-639)
The constraint on the Y-axis are(0-479)
Enter the coordinates of the clipping window (xmin): 50
Enter the coordinates of the clipping window (ymin): 60
Enter the coordinates of the clipping window (xmax): 300
Enter the coordinates of the clipping window (ymax): 240
Enter the coordinates of the line to be clipped (x1): 20
Enter the coordinates of the line to be clipped (y1): 39
Enter the coordinates of the line to be clipped (x2): 550
Enter the coordinates of the line to be clipped (y2): 400
```



```
dy;

y22);
"(x11, y11)");
"(x22, y22)");

);
n the X-axis are(0-%d)\n", getmaxx());
n the Y-axis are(0-%d)\n", getmaxy());

ipping window
inates of the clipping window (xmin): ";

inates of the clipping window (ymin): ";
```

```
C:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode\Home\build\1
1_LiangBarskyLineClipping.exe"
The constraint on the X-axis are(0-639)
The constraint on the Y-axis are(0-479)
Enter the coordinates of the clipping window (xmin): 50
Enter the coordinates of the clipping window (ymin): 60
Enter the coordinates of the clipping window (xmax): 300
Enter the coordinates of the clipping window (ymax): 240
Enter the coordinates of the line to be clipped (x1): 20
Enter the coordinates of the line to be clipped (y1): 39
Enter the coordinates of the line to be clipped (x2): 550
Enter the coordinates of the line to be clipped (y2): 400
```

NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

PRACTICLE-12

OBJECTIVE- To implement 2D Transformations on a triangle: translation, rotation and scaling.

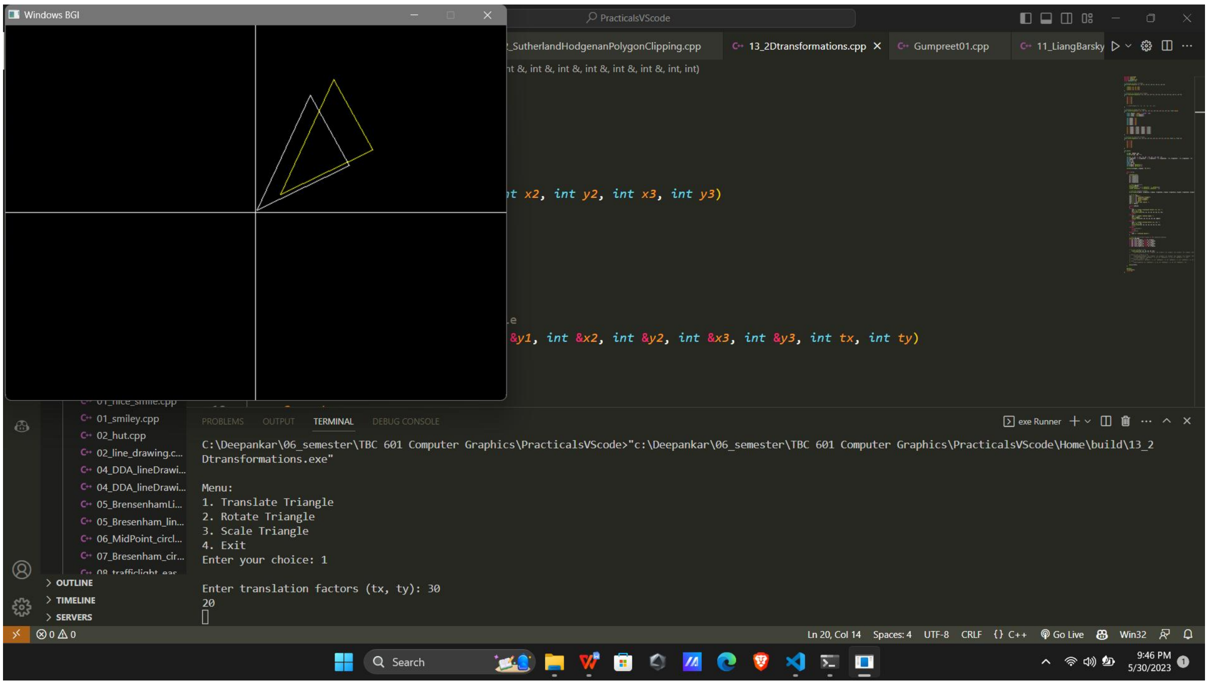
SYNTAX:-

```
#include <iostream>
#include <math.h>
#include <graphics.h>
using namespace std;

// Function to draw a triangle
void drawTriangle(int x1, int y1, int x2, int y2, int x3, int y3)
{
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
}

// Function to translate the triangle
void translateTriangle(int &x1, int &y1, int &x2, int &y2, int &x3,
int &y3, int tx, int ty)
{
    x1 += tx;
    y1 += ty;
    x2 += tx;
    y2 += ty;
    x3 += tx;
    y3 += ty;

    // drawTriangle( x1,  y1,  x2,  y2,  x3,  y3);
}

// Function to rotate the triangle
void rotateTriangle(int &x1, int &y1, int &x2, int &y2, int &x3, int
&y3, float angle)
{
    float radians = angle * 3.14159 / 180;
    float cosVal = cos(radians);
    float sinVal = sin(radians);

    int tempX1 = x1;
    int tempX2 = x2;
    int tempX3 = x3;
    int tempY1 = y1;
    int tempY2 = y2;
    int tempY3 = y3;

    x1 = tempX1 * cosVal - tempY1 * sinVal;
    y1 = tempX1 * sinVal + tempY1 * cosVal;
    x2 = tempX2 * cosVal - tempY2 * sinVal;
    y2 = tempX2 * sinVal + tempY2 * cosVal;
    x3 = tempX3 * cosVal - tempY3 * sinVal;
    y3 = tempX3 * sinVal + tempY3 * cosVal;
}

// Function to scale the triangle
void scaleTriangle(int &x1, int &y1, int &x2, int &y2, int &x3, int
&y3, float sx, float sy)
{
    x1 *= sx;
    y1 *= sy;
    x2 *= sx;
    y2 *= sy;
    x3 *= sx;
    y3 *= sy;
}

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int x1 = 2, y1 = 3, x2 = 70, y2 = 150, x3 = 120, y3 = 60;
    int originalx1 = 2, originaly1 = 3, originalx2 = 70, originaly2
= 150, originalx3 = 120, originaly3 = 60;
    int choice;
    int tx, ty;
    float angle;
    float sx, sy;
    int originx= getmaxx()/2;
    int originy= getmaxy()/2;

    outtextxy(originx, originy, "(0, 0)");


    while (true)
    {
        cleardevice();
        x1= originalx1;
        x2= originalx2;
        x3= originalx3;
        y1= originaly1;
        y2= originaly2;
        y3= originaly3;
        // Draw the quadrants
        setcolor(WHITE);
        line(getmaxx() / 2, 0, getmaxx() / 2, getmaxy());
        line(0, getmaxy() / 2, getmaxx(), getmaxy() / 2);

        // Draw the original triangle
        drawTriangle(originx+ originalx1, originy- originaly1,
originx+ originalx2, originy- originaly2, originx+ originalx3,
originy- originaly3);

        // Print menu
        cout << "\nMenu:";
        cout << "\n1. Translate Triangle";
        cout << "\n2. Rotate Triangle";
        cout << "\n3. Scale Triangle";
        cout << "\n4. Exit";
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            cout << "\nEnter translation factors (tx, ty): ";
            cin >> tx >> ty;
            translateTriangle(x1, y1, x2, y2, x3, y3, tx, ty);
            break;
        case 2:
            cout << "\nEnter rotation angle: ";
            cin >> angle;
            rotateTriangle(x1, y1, x2, y2, x3, y3, angle);
            break;
        case 3:
            cout << "\nEnter scaling factors (sx, sy): ";
            cin >> sx >> sy;
            scaleTriangle(x1, y1, x2, y2, x3, y3, sx, sy);
            break;
        case 4:
            // closegraph();
            break;
            // exit(0);
        default:
            cout << "\nInvalid choice!";
        }

        // Draw the transformed triangle in the respective quadrant
        setcolor(YELLOW);
        if (x1>=0)x1+=originx; else x1-=originx;
        if (y1>=0)y1=originy-y1; else y1+=originy;
        if (x2>=0)x2+=originx; else x2-=originx;
        if (y2>=0)y2=originy-y2; else y2+=originy;
        if (x3>=0)x3+=originx; else x3-=originx;
        if (y3>=0)y3=originy-y3; else y3+=originy;
        // if (x1 >= 0 && y1 >= 0)
            drawTriangle(x1, y1, x2, y2, x3, y3);
        //     drawTriangle(originx + x1, originy - y1, originx + x2,
originy - y2, originx + x3, originy - y3);

        // else if (x1 < 0 && y1 >= 0)
        //     drawTriangle(originx + x1, originy - y1, originx + x2,
originy - y2, originx + x3, originy - y3);
        //     // drawTriangle(x1 + getmaxx() / 2, y1, x2 + getmaxx()
/ 2, y2, x3 + getmaxx() / 2, y3);
        // else if (x1 < 0 && y1 < 0)
        //     drawTriangle(x1 + getmaxx() / 2, y1 + getmaxy() / 2,
x2 + getmaxx() / 2, y2 + getmaxy() / 2, x3 + getmaxx() / 2, y3 +
getmaxy() / 2);
        // else
        //     drawTriangle(x1, y1 + getmaxy() / 2, x2, y2 + getmaxy()
/ 2, x3, y3 + getmaxy() / 2);

        delay(10000);
    }

    getch();
    closegraph();
    return 0;
}
```

OUTPUT:-

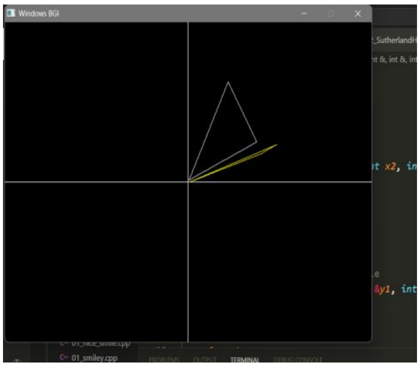
```
C:\Deepankar\06_semester\TBC 601 Computer
Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer
Graphics\PracticalsVScode\Home\build\13_2Dtransformations.exe"

Menu:
1. Translate Triangle
2. Rotate Triangle
3. Scale Triangle
4. Exit
Enter your choice: 1

Enter translation factors (tx, ty): 30
20

Menu:
1. Translate Triangle
2. Rotate Triangle
3. Scale Triangle
4. Exit
Enter your choice: 2

Enter rotation angle: -45
```

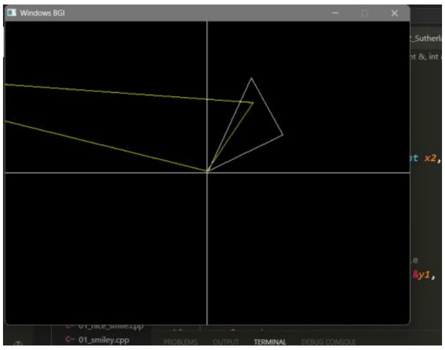
```
Menu:
1. Translate Triangle
2. Rotate Triangle
3. Scale Triangle
4. Exit
Enter your choice: 2

Enter rotation angle: 30

Menu:
1. Translate Triangle
2. Rotate Triangle
3. Scale Triangle
4. Exit
Enter your choice: 3

Enter scaling factors (sx, sy): 3
4

Menu:
1. Translate Triangle
2. Rotate Triangle
3. Scale Triangle
4. Exit
Enter your choice: 4
```

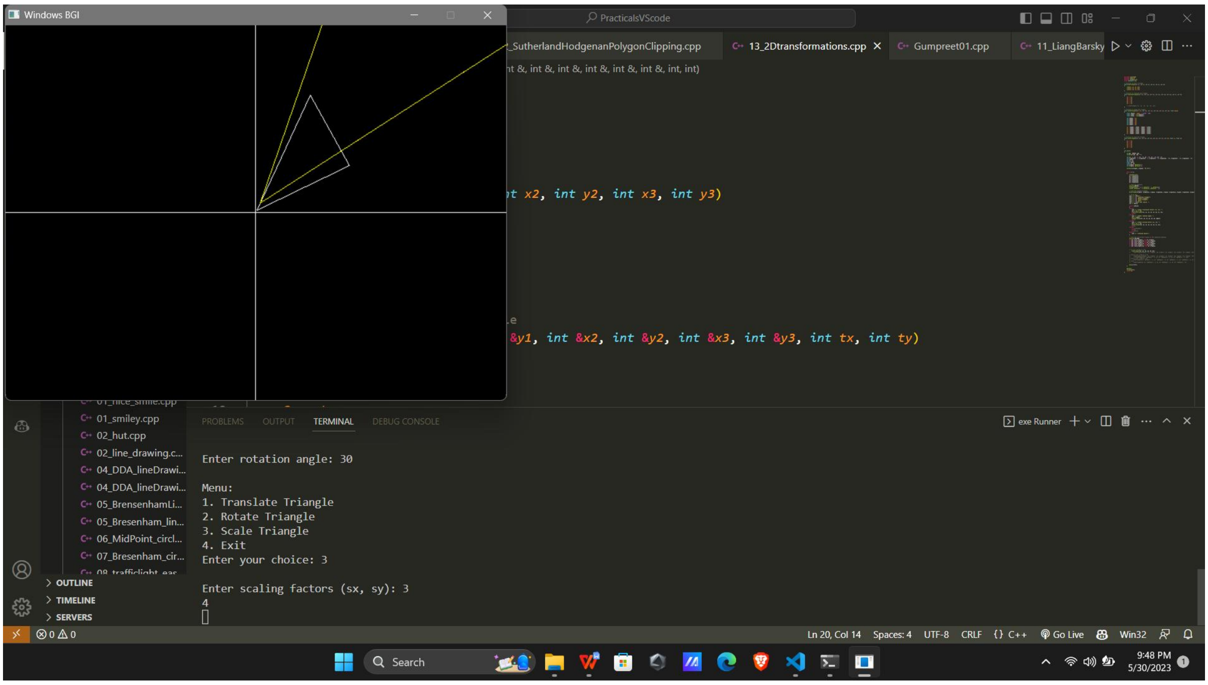


```
t &, int &, int &, int &, int &, int &, int, int)
t x2, int y2, int x3, int y3)
&y1, int &x2, int &y2, int &x3, int &y3, int tx, int ty)
```

```
C:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode\Home\build\13_2
Dtransformations.exe"

Menu:
1. Translate Triangle
2. Rotate Triangle
3. Scale Triangle
4. Exit
Enter your choice: 1

Enter translation factors (tx, ty): 30
20
```







```
nt &, int &, int &, int &, int &, int &, int, int)

nt x2, int y2, int x3, int y3)

&y1, int &x2, int &y2, int &x3, int &y3, int tx, int ty)
```

```
Enter rotation angle: 30

Menu:
1. Translate Triangle
2. Rotate Triangle
3. Scale Triangle
4. Exit
Enter your choice: 3

Enter scaling factors (sx, sy): 3
4
```

NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

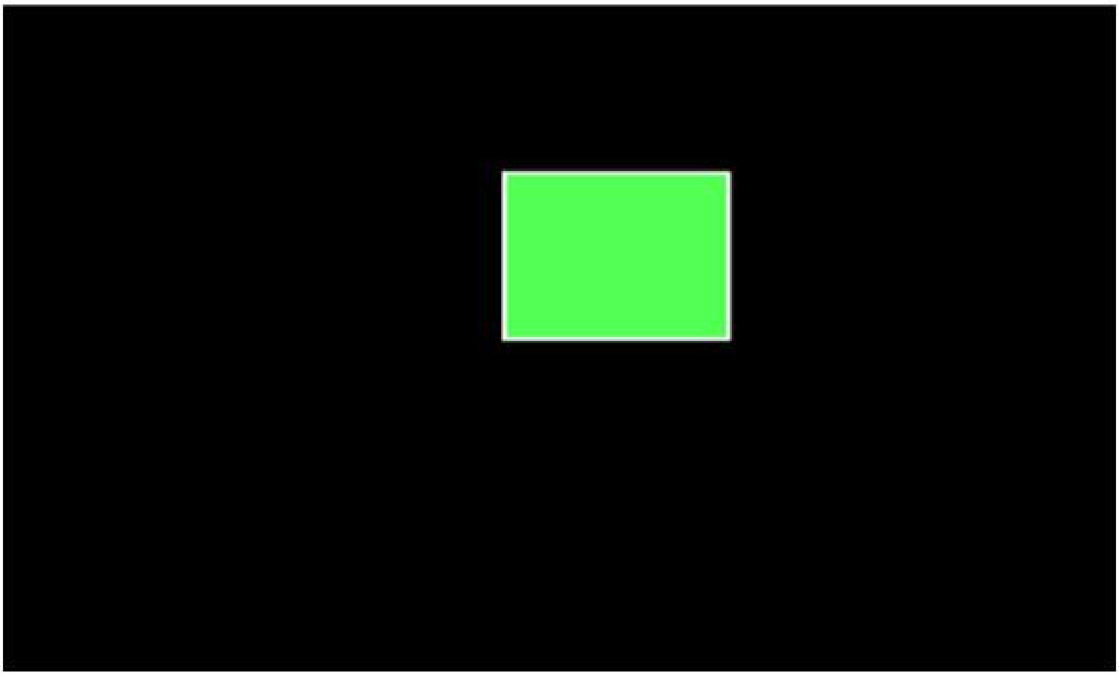
PRACTICLE-13

OBJECTIVE- To implement Flood Fill Algorithm through graphics.

SYNTAX:-

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void flood(int,int,int,int);
void main()
{
    intgd=DETECT,gm;
    initgraph(&gd,&gm,"C:/TURBOC3/bgi");
    rectangle(50,50,250,250);
    flood(55,55,10,0);
    getch();
}
void flood(intx,inty,intfillColor, intdefaultColor)
{
    if(getpixel(x,y)==defaultColor)
    {
        delay(1);
        putpixel(x,y,fillColor);
        flood(x+1,y,fillColor,defaultColor);
        flood(x-1,y,fillColor,defaultColor);
        flood(x,y+1,fillColor,defaultColor);
             flood(x,y-1,fillColor,defaultColor);
    }
   }
```

OUTPUT:-



NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

PRACTICLE-13

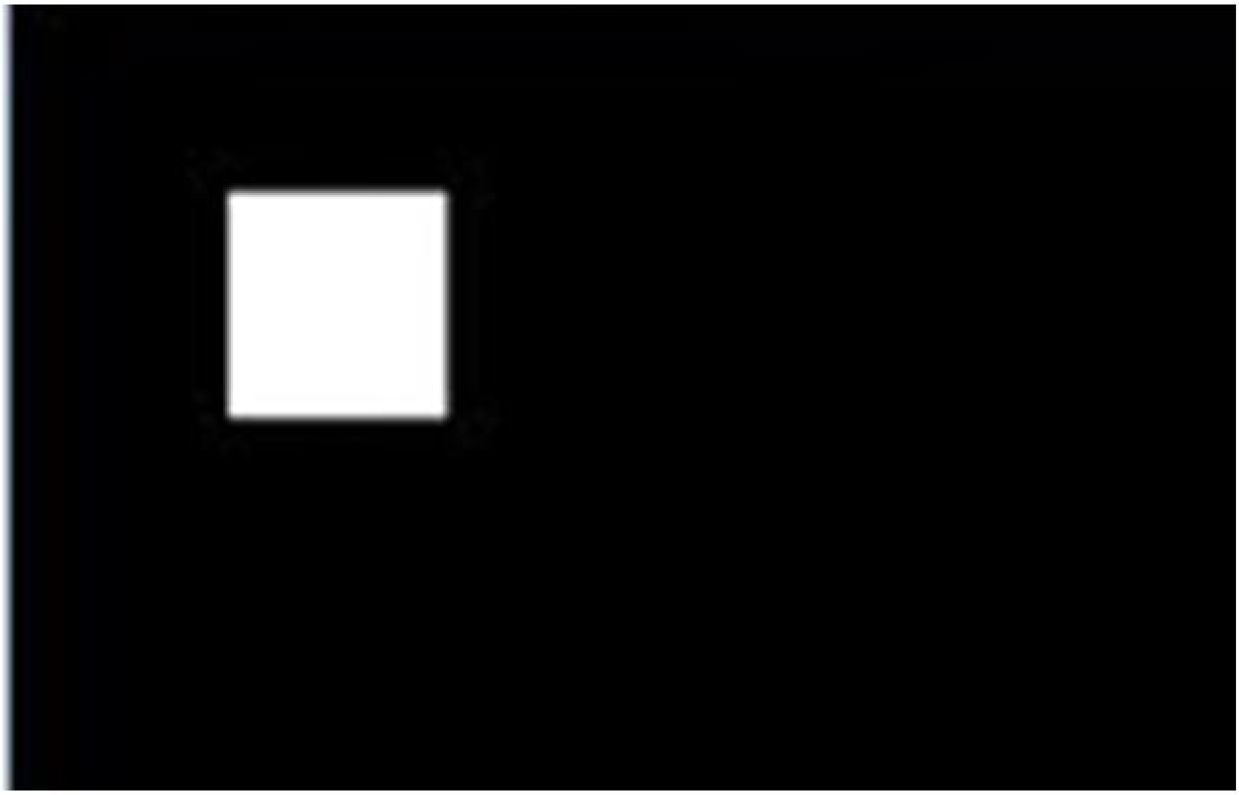
OBJECTIVE- To implement 8-connected Flood Fill Algorithm through graphics.

SYNTAX:-

```
#include<stdio.h>
#include<graphics.h>
#include<dos.h>
#include<conio.h>
void floodfill(intx,inty,intold,intnewcol)
{
                int current;
                current=getpixel(x,y);
                if(current==old)
                 {
                                delay(5);
                                putpixel(x,y,newcol);
                                floodfill(x+1,y,old,newcol);
                                floodfill(x-1,y,old,newcol);
                                floodfill(x,y+1,old,newcol);
```

```
                        floodfill(x,y-1,old,newcol);
                        floodfill(x+1,y+1,old,newcol);
                        floodfill(x-1,y+1,old,newcol);
                        floodfill(x+1,y-1,old,newcol);
                        floodfill(x-1,y-1,old,newcol);
        }
}
void main()
{
        intgd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
        rectangle(50,50,150,150);
        floodfill(70,70,0,15);
        getch();
        closegraph();
}
```

Output:



NAME- Deepankar Sharma
COURSE- BCA
ROLL NO- 2092014
SUBJECT- Computer graphics lab

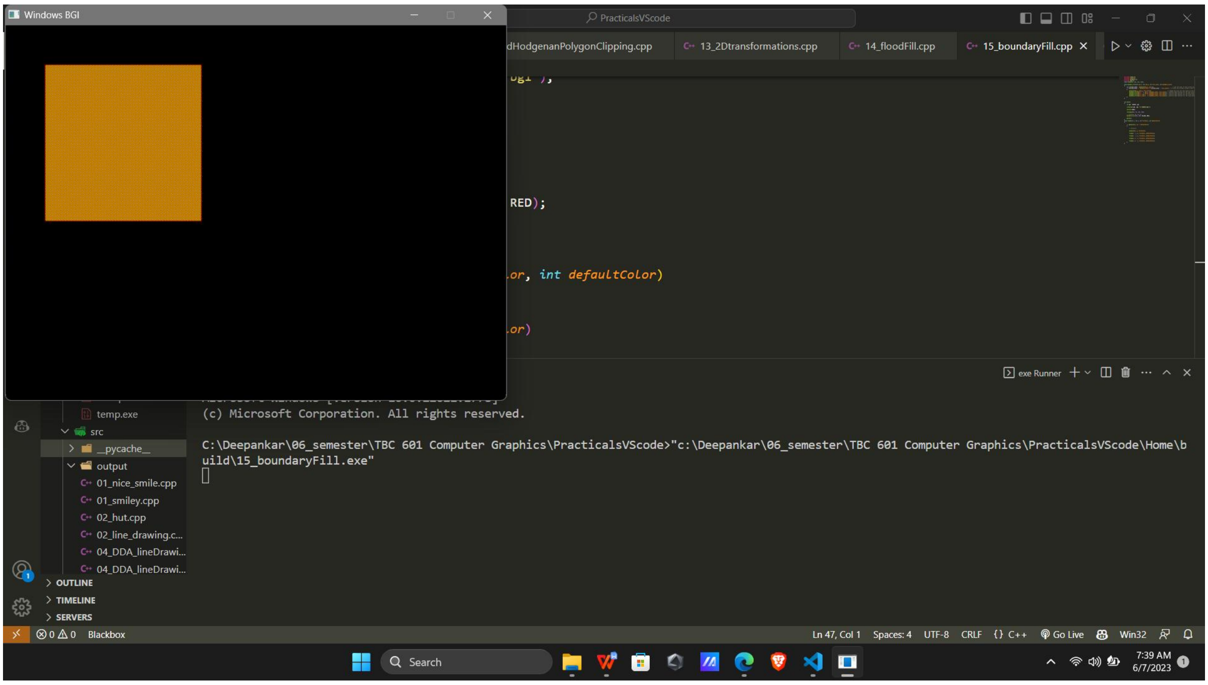
PRACTICLE-14

OBJECTIVE- To implement Boundary Fill Algorithm through graphics.

SYNTAX:-

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>
void flood(int, int, int, int);

void boundary_fill(int pos_x, int pos_y, int fill_color, int
boundary_color)
{
    int current_color = getpixel(pos_x, pos_y);
// get the color of the current pixel position
    if (current_color != boundary_color && current_color !=
fill_color) // if pixel not already filled or part of the boundary
then
    {
        putpixel(pos_x, pos_y, fill_color);                        //
change the color for this pixel to the desired fill_color
        boundary_fill(pos_x + 1, pos_y, boundary_color, fill_color);
// perform same function for the east pixel
        boundary_fill(pos_x - 1, pos_y, boundary_color, fill_color);
// perform same function for the west pixel
        boundary_fill(pos_x, pos_y + 1, boundary_color, fill_color);
// perform same function for the north pixel
        boundary_fill(pos_x, pos_y - 1, boundary_color, fill_color);
// perform same function for the south pixel
    }
}


int main()
{
    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:/TURBOC3/bgi");

    setcolor(RED);

    rectangle(50, 50, 250, 250);

    // flood(55, 55, 10, 0);
    boundary_fill(105, 200, YELLOW, RED);

    getch();
}
void flood(int x, int y, int fillColor, int defaultColor)
{

    if (getpixel(x, y) == defaultColor)
    {

        // delay(1);

        putpixel(x, y, fillColor);

        flood(x + 1, y, fillColor, defaultColor);

        flood(x - 1, y, fillColor, defaultColor);

        flood(x, y + 1, fillColor, defaultColor);

        flood(x, y - 1, fillColor, defaultColor);
    }
}
```

Output:



```
(c) Microsoft Corporation. All rights reserved.

C:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode>"c:\Deepankar\06_semester\TBC 601 Computer Graphics\PracticalsVScode\Home\b
uild\15_boundaryFill.exe"
```