NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

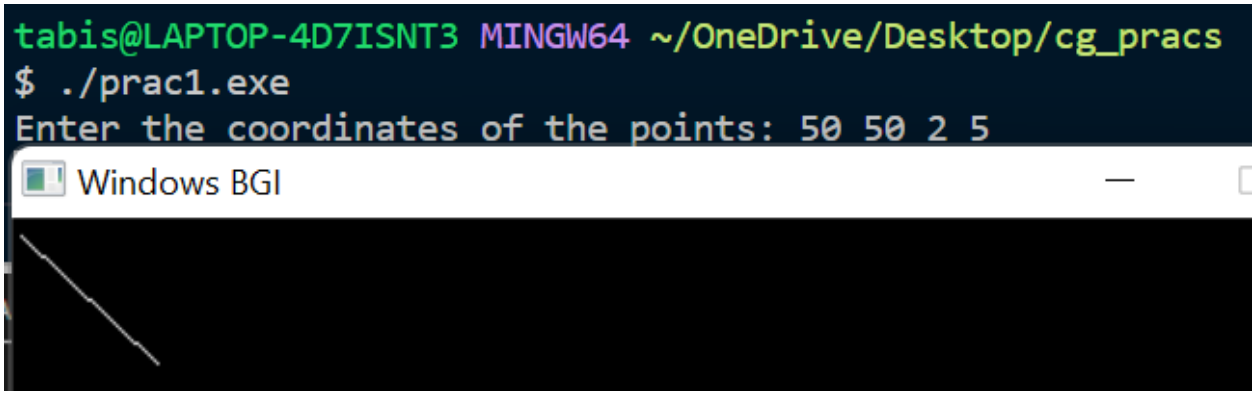
Computer Graphics (COCSE64)
Practical File

Submitted by :
Tabishi Singh
(2019UCO1506)

Exp no.	Experiment Name
1.	Generating line primitives using DDA
2.	Generating line primitives using Bresenham's Approach
3.	Generating circle using bresenham's approach
4.	Generating circle using mid point algorithm
5.	Generating ellipse using mid-point approach
6.	Generating hyperbola using mid point algorithm
7.	Implement line clipping approach using cohen sutherland
8.	Implement line clipping approach using liang barsky / cyrus beck
9.	Implement line clipping approach using mid-point subdivision

1 .Generating line primitives using DDA

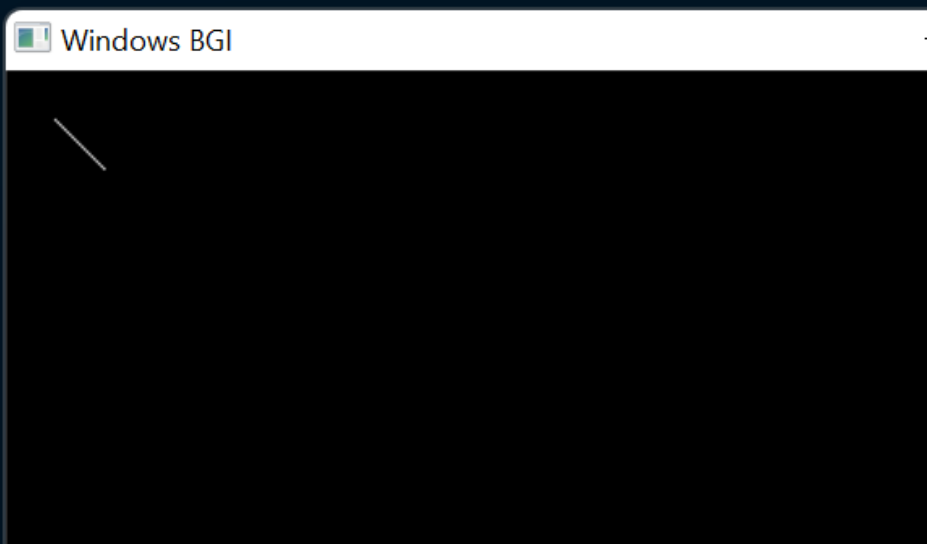
```
#include <iostream>
#include <graphics.h>
#include <cmath>
#include <time.h>
using namespace std;
//function to generate the line
void DDALine(int x0, int y0, int x1, int y1){
    int dx = x1 - x0;
    int dy = y1 - y0;

    int step = (abs(dx) > abs(dy))? abs(dx) : abs(dy);

    float x_step = (float)dx/step;
    float y_step = (float)dy/step;

    float x = x0;
    float y = y0;

    for(int i = 0; i < step; i++){
        putpixel(round(x), round(y), WHITE);
        // cout << round(x) << " " << round(y) << endl;
        x += x_step;
        y += y_step;
        delay(10);
    }

}

//driver function
int main(){
    initwindow(500,500);
    int x0, y0, x1, y1;
    cout << "Enter the coordinates of the points: ";
    cin >> x0 >> y0 >> x1 >> y1;

    DDALine(x0, y0, x1, y1);

    delay(100);
    getch();
    delay(10000);
    closegraph();
    return 0;
}
```


tabis@LAPTOP-4D7ISNT3 MINGW64 ~/OneDrive/Desktop/cg_pracs
$ ./prac1.exe
Enter the coordinates of the points: 50 50 2 5
Windows BGI

2 . Generating Line using Bresenham's Approach

```
#include <bits/stdc++.h>
#include <graphics.h>
using namespace std;

// function for line generation
void bresenham(int x1, int y1, int x2, int y2)
{
    int m_new = 2 * (y2 - y1);
    int slope_error_new = m_new - (x2 - x1);

    for (int x = x1, y = y1; x <= x2; x++)
    {
        putpixel(x, y, WHITE);
        cout << x << " " << y << endl;
        slope_error_new += m_new;
        if (slope_error_new >= 0)
        {
            y++;
            slope_error_new -= 2 * (x2 - x1);
        }
        delay(10);
    }
}

int main()
{
    initwindow(500, 500);
    int x1, y1, x2, y2;
    cout << "Enter the coordinates of the points: ";
    cin >> x1 >> y1 >> x2 >> y2;

    bresenham(x1, y1, x2, y2);
    delay(100);
    getch();
    closegraph();
    return 0;
}
```

```
tabis@LAPTOP-4D7ISNT3 MINGW64 ~/OneDrive/Desktop/cg_pracs
$ ./prac2.exe
Enter the coordinates of the points: 20 20
40 50
28 28
29 29
30 30
31 31
32 32
33 33
34 34
35 35
36 36
37 37
38 38
39 39
40 40
```



3. Generating circle using bresenham's approach

```
#include <bits/stdc++.h>
using namespace std;
#include <graphics.h>
void draw(int x, int y)
{
    putpixel(x + 200, y + 200, WHITE);
    delay(1);
    putpixel(x + 200, -y + 200, WHITE);
    delay(1);
    putpixel(-x + 200, -y + 200, WHITE);
    delay(1);
    putpixel(-x + 200, y + 200, WHITE);
    delay(1);
    putpixel(y + 200, x + 200, WHITE);
    delay(1);
    putpixel(y + 200, -x + 200, WHITE);
    delay(1);
    putpixel(-y + 200, x + 200, WHITE);
    delay(1);
    putpixel(-y + 200, -x + 200, WHITE);
}
void circle1(int a, int b, int c)
{
    float p = 3 - 2 * c;
    int x = 0, y = c;
    cout << x << ", " << y << "\n";
    putpixel(x + a, y + b, WHITE);
    while (x <= y)
    {
        if (p < 0)
        {
            p = p + 4 * x + 6;
            x = x + 1;
            draw(x, y);
        }
        else
        {
```

```
            p = p + 4 * (x - y) + 10;
            x = x + 1;
            y = y - 1;
            draw(x, y);
        }
    }
}
int main()
{
    initwindow(720, 720);
    int a = 150, b = 150, c = 100;
    circle1(a, b, c);
    getch();
    closegraph();
    return 0;
}
```

```
tabis@LAPTOP-4D7ISNT3 MINGW64 ~/OneDrive/Desktop/cg_pracs
$ ./prac3.exe
0, 100
```

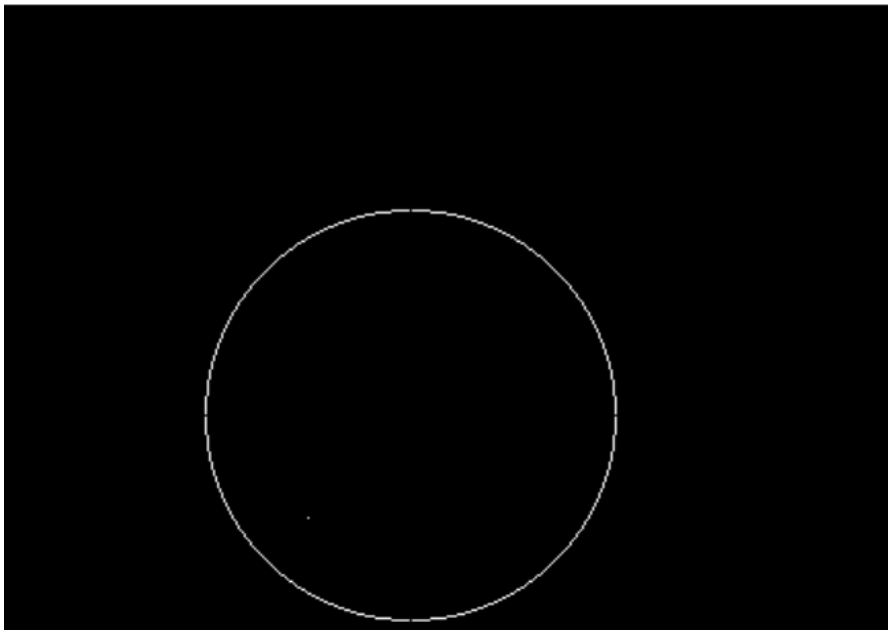
4. Generating circle using Mid point approach

```
using namespace std;
#include <iostream>
#include <graphics.h>
void draw(int x, int y)
{
    putpixel(x + 200, y + 200, WHITE);
    delay(1);
    putpixel(x + 200, -y + 200, WHITE);
    delay(1);
    putpixel(-x + 200, -y + 200, WHITE);
    delay(1);
    putpixel(-x + 200, y + 200, WHITE);
    delay(1);
    putpixel(y + 200, x + 200, WHITE);
    delay(1);
    putpixel(y + 200, -x + 200, WHITE);
    delay(1);
    putpixel(-y + 200, x + 200, WHITE);
    delay(1);
    putpixel(-y + 200, -x + 200, WHITE);
}
void circle1(int a, int b, int c)
{
    float p = 3 - 2 * c;
    int x = 0, y = c;
    cout << x << ", " << y << "\n";
    putpixel(x + a, y + b, WHITE);
    while (x <= y)
    {
        if (p < 0)
        {
            p = p + 4 * x + 6;
            x = x + 1;
            draw(x, y);
        }
        else
        {
```

```
            p = p + 4 * (x - y) + 10;
            x = x + 1;
            y = y - 1;
            draw(x, y);
        }
    }
}
int main()
{
    initwindow(720, 720);
    int a = 150, b = 150, c = 100;
    circle1(a, b, c);
    getch();
    closegraph();
    return 0;
}
```

```
tabis@LAPTOP-4D7ISNT3 MINGW64 ~/OneDrive/Desktop/cg_pracs
$ ./prac4.exe
0, 100
```

Windows BGI



5. Generating ellipse using mid point algorithm

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
void disp();
float x, y;
int xc, yc;
int main()
{
    float p1, p2;
    initwindow(720, 720);
    int a, b;
    printf("*** Ellipse Generating Algorithm ***\n");
    printf("Enter the value of Xc\t");
    scanf("%d", &xc);
    printf("Enter the value of yc\t");
    scanf("%d", &yc);
    printf("Enter X axis length\t");
    scanf("%d", &a);
    printf("Enter Y axis length\t");
    scanf("%d", &b);
    x = 0;
    y = b;
    disp();
    p1 = (b * b) - (a * a * b) + (a * a) / 4;
    while ((2.0 * b * b * x) <= (2.0 * a * a * y))
    {
        x++;
        if (p1 <= 0)
            p1 = p1 + (2.0 * b * b * x) + (b * b);
        else
        {
            y--;
            p1 = p1 + (2.0 * b * b * x) + (b * b) - (2.0 * a * a * y);
        }
```

```
        disp();
        x = -x;
        disp();
        x = -x;
        delay(50);
    }
    x = a;
    y = 0;
    disp();
    p2 = (a * a) + 2.0 * (b * b * a) + (b * b) / 4;
    while ((2.0 * b * b * x) > (2.0 * a * a * y))
    {
        y++;
        if (p2 > 0)
            p2 = p2 + (a * a) - (2.0 * a * a * y);
        else
        {
            x--;
            p2 = p2 + (2.0 * b * b * x) - (2.0 * a * a * y) + (a * a);
        }
        disp();
        y = -y;
        disp();
        y = -y;
        delay(50);
    }
    getch();
    closegraph();
    return 0;
}
void disp()
{
    putpixel(xc + x, yc + y, 7);
    putpixel(xc - x, yc + y, 7);
    putpixel(xc + x, yc - y, 7);
    putpixel(xc + x, yc - y, 7);
}
```

```
tabis@LAPTOP-4D7ISNT3 MINGW64 ~/OneDrive/Desktop/cg_pracs
$ ./prac5.exe
*** Ellipse Generating Algorithm ***
Enter the value of Xc   50
Enter the value of yc   60
Enter X axis length     50
Enter Y axis length     30
```

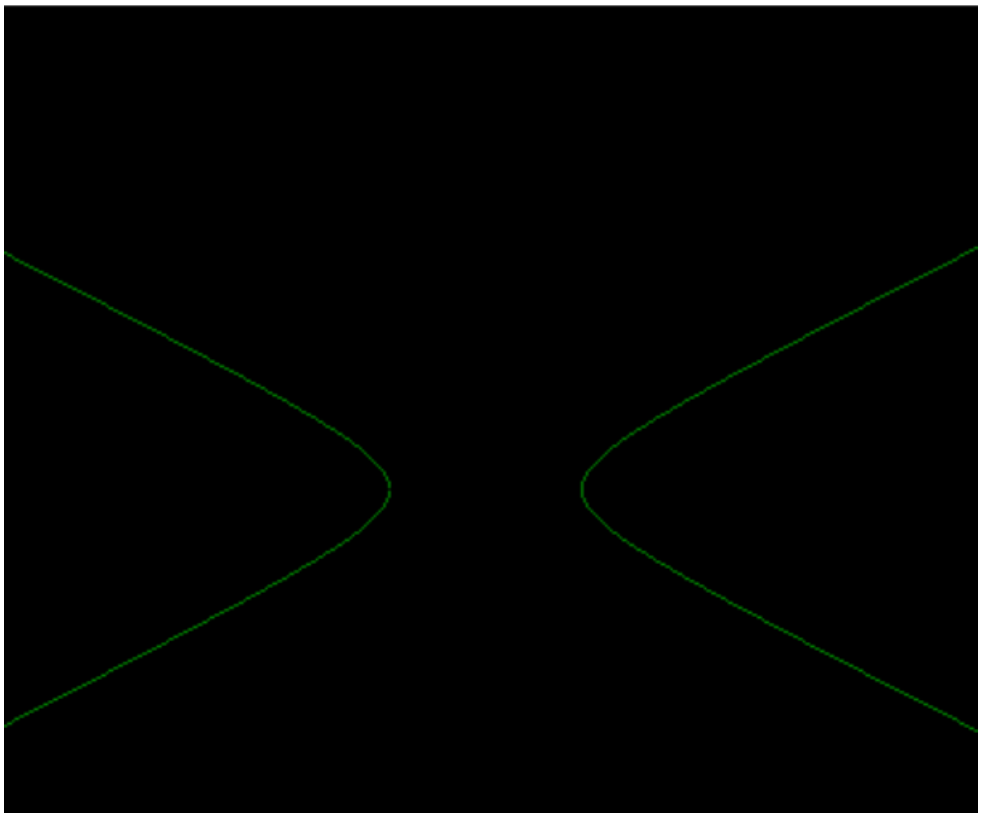
Windows BGI

6. Generating hyperbola using Mid point algorithm

```
#include <bits/stdc++.h>
using namespace std;
#include <graphics.h>
// // Driver program
void draw(int x, int y)
{
    putpixel(x + 200, y + 200, GREEN);
    delay(1);
    putpixel(x + 200, -y + 200, GREEN);
    delay(1);
    putpixel(-x + 200, -y + 200, GREEN);
    delay(1);
    putpixel(-x + 200, y + 200, GREEN);
    delay(1);
}
void hyp(int a, int b)
{
    double p = (float)(1 / 4 + a) * (b * b) - (float)1 * (a * a);
    cout << p << "\n";
    int x = a, y = 0;
    putpixel(x + 200, y + 200, GREEN);
    while (y < (b * b) / (sqrt(a * a - b * b)))
    {
        if (p > 0)
        {
            p = p - (2 * y + 3) * (a * a);
            y = y + 1;
            draw(x, y);
        }
        else
        {
            p = p + (2 * (x + 1) * (b * b)) - (2 * y + 3) * (a * a);
            y = y + 1;
            x = x + 1;
            draw(x, y);
```

```
        }
        cout << p << "\n";
    }
    p = (x + 1) * (x + 1) * b * b - (y + 1 / 2) * (y + 1 / 2) * a * a - a * a * b *
b;
    while (y < 300)
    {
        cout << "aa";
        if (p > 0)
        {
            p = p + (2 * x + 3) * b * b - a * a * (2 * (y + 1));
            y = y + 1;
            x = x + 1;
            draw(x, y);
        }
        else
        {
            p = p + (2 * x + 3) * b * b;
            x = x + 1;
            draw(x, y);
        }
    }
}
int main()
{
    initwindow(1000, 1000);
    int a = 40, b = 20;
    hyp(a, b);
    getch();
    return 0;
}
```

Windows BGI



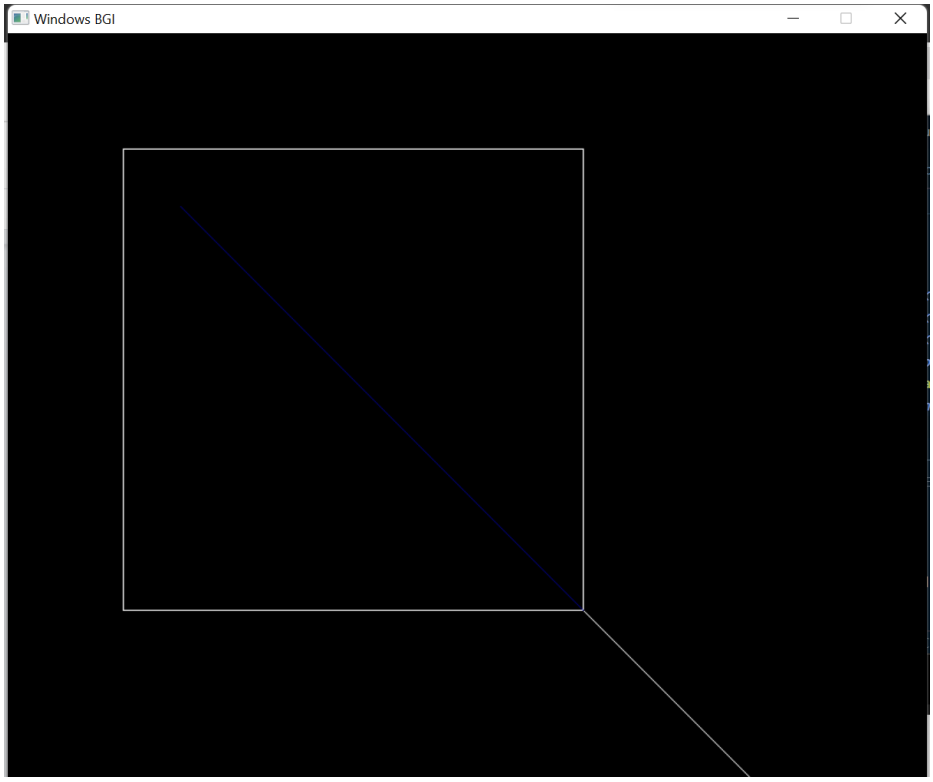
7. Implement Line Clipping approach using Cohen Sutherland

```
#include <bits/stdc++.h>
#include <graphics.h>

using namespace std;
int main()
{
    int t;
    initwindow(800, 800);
    double xmin, xmax, ymin, ymax;
    cout << "Enter xmin xmax ymin ymax\n";
    cin >> xmin >> xmax >> ymin >> ymax;
    rectangle(xmin, ymin, xmax, ymax);
    cout << "No of lines:";
    cin >> t;
    while (t--)
    {
        double x1, y1, x2, y2;
        cout << "Enter x1,y1 x2,y2\n";
        cin >> x1 >> y1 >> x2 >> y2;
        line(x1, y1, x2, y2);
        double slope = (y2 - y1) / (x2 - x1);
        int arr1[4] = {0}, arr2[4] = {0}; // TBRL
        if (x1 > xmax)
            arr1[2] = 1; // Right
        if (x1 < xmin)
            arr1[3] = 1; // Left
        if (y1 > ymax)
            arr1[0] = 1; // Top
        if (y1 < ymin)
            arr1[1] = 1; // Bottom
        if (x2 > xmax)
            arr2[2] = 1;
        if (x2 < xmin)
            arr2[3] = 1;
```

```
        if (y2 > ymax)
            arr2[0] = 1;
        if (y2 < ymin)
            arr2[1] = 1;
        cout << "Region codes are\n";
        cout << "For x1,y1\n";
        for (int i = 0; i < 4; i++)
            cout << arr1[i];
        cout << "\nFor x2,y2\n";
        for (int i = 0; i < 4; i++)
            cout << arr2[i];
        int flag = 1;
        for (int i = 0; i < 4; i++)
        {
            if ((arr1[i] & arr2[i]) == 1)
            {
                cout << "\nClipping not possible\n";
                flag = 0;

                break;
            }
        }
        if (flag)
        {
            cout << endl;
            if (arr1[2])
            {
                y1 += slope * (xmax - x1);
                x1 = xmax;
            }
            if (arr1[3])
            {
                y1 += slope * (xmin - x1);
                x1 = xmin;
            }
            if (arr1[0])
            {
                x1 += (ymax - y1) / slope;
```

```
                y1 = ymax;
            }
            if (arr1[1])
            {
                x1 += (ymin - y1) / slope;
                y1 = ymin;
            }
            if (arr2[2])
            {
                y2 += slope * (xmax - x2);
                x2 = xmax;
            }
            if (arr2[3])
            {
                y2 += slope * (xmin - x2);
                x2 = xmin;
            }
            if (arr2[0])
            {
                x2 += (ymax - y2) / slope;
                y2 = ymax;
            }
            if (arr2[1])
            {
                x2 += (ymin - y2) / slope;
                y2 = ymin;
            }
        }
        cout << "Coordinates of clipped lines are:";
        cout << "(" << x1 << "," << y1 << ")";
        cout << "(" << x2 << "," << y2 << ")" << endl;
        setcolor(1);
        if (flag)
            line(x1, y1, x2, y2);
    }
    getch();
    closegraph();
}
```

```
tabis@LAPTOP-4D7ISNT3 MINGW64 ~/OneDrive/Desktop/cg_pracs
$ ./prac7.exe
Enter xmin xmax ymin ymax
100 500 100 500
No of lines:1
Enter x1,y1 x2,y2
150 150 700 700
Region codes are
For x1,y1
0000
For x2,y2
1010
Coordinates of clipped lines are:(150,150)(500,500)
```

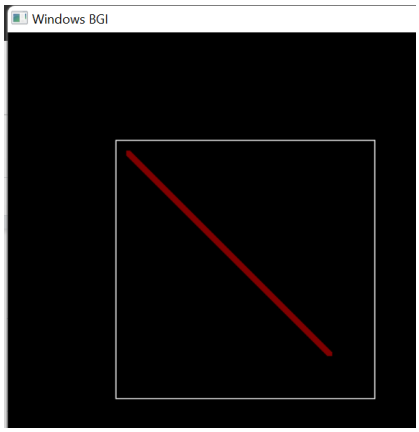


8. Implement Line Clipping approach using Liang Barsky / Cyrus Beck

```
#include <bits/stdc++.h>
#include <graphics.h>

using namespace std;
int xy[4];

void LiangBarsky(float x1, float y1, float x2, float y2) {
  float p[5], q[5];  // left-1 right-2 bottom-3 top-4
  p[1] = x1 - x2, p[2] = x2 - x1, p[3] = y1 - y2, p[4] = y2 - y1;
  q[1] = x1 - xy[0], q[2] = xy[1] - x1, q[3] = y1 - xy[2], q[4] = xy[3] - y1;
  float u1 = 0, u2 = 1;
  for (int i = 1; i < 5; i++) {
    if (p[i] < 0) {
      float r = q[i] / p[i];
      u1 = max(r, u1);
    }
    if (p[i] > 0) {
      float r = q[i] / p[i];
      u2 = min(u2, r);
    }
  }
  if (u1 > u2) {
    cout << "Line is out of clipping window";
    return;
  }
  cout << u1 << " " << u2 << endl;
  float x11 = x1 + u1 * (x2 - x1);
  float x12 = x1 + u2 * (x2 - x1);
  float y11 = y1 + u1 * (y2 - y1);
  float y12 = y1 + u2 * (y2 - y1);
  cout << "\nClipped Line is: (" << x11 << "," << y11 << ") , (" << x12 << "," << y12
<< ")\n\n";
  setcolor(4);
  setlinestyle(0, 0, 6);
  line(x11, y11, x12, y12);
}
```

```
int main() {
  initwindow(800, 800);
  cout << "Enter xmin xmax ymin ymax" << endl;
  cin >> xy[0] >> xy[1] >> xy[2] >> xy[3];
  rectangle(xy[0], xy[2], xy[1], xy[3]);
  cout << "Enter Line Coordinates" << endl;
  float x1, y1, x2, y2;
  cin >> x1 >> y1 >> x2 >> y2;
  setcolor(1);
  setlinestyle(0, 0, 4);
  line(x1, y1, x2, y2);
  LiangBarsky(x1, y1, x2, y2);
  getch();
  return 0;
}
```

```
tabis@LAPTOP-4D7ISNT3 MINGW64 ~/OneDrive/Desktop/cg_pracs
$ ./prac8.exe
Enter xmin xmax ymin ymax
100 340 100 340
Enter Line Coordinates
110 110
300 300
0 1

Clipped Line is: (110,110) , (300,300)
```



9. Implement Line Clipping approach using Mid-Point Subdivision

```
#include <bits/stdc++.h>
#include <graphics.h>
using namespace std;
#define XWMIN 100
#define XWMAX 500
#define YWMIN 100
#define YWMAX 500

int calcCode(int x, int y)
{
    int code = 0;
    if (x < XWMIN) // left
        code |= 1;
    else if (x > XWMAX) // right
        code |= 2;
    if (y < YWMIN) // top
        code |= 4;
    else if (y > YWMAX) // bottom
        code |= 8;
    return code;
}
void clipLine(int &xc1, int &yc1, int &xc2, int &yc2, int x1, int y1, int x2, int y2)
{
    int xc11, yc11, xc12, yc12, xc21, yc21, xc22, yc22;
    int code1 = calcCode(x1, y1), code2 = calcCode(x2, y2);
    if (x1 == (x1 + x2) / 2 && y1 == (y1 + y2) / 2)
    {
        xc1 = x1;
        xc2 = x2;
        yc1 = y1;
        yc2 = y2;
        return;
    }
    if ((code1 | code2) == 0)
    {
        // completely inside
```

```
        xc1 = x1;
        yc1 = y1;
        xc2 = x2;
        yc2 = y2;
        return;
    }
    else if ((code1 & code2) != 0)
    {
        // completely outside
        xc1 = -1;
        yc1 = -1;
        xc2 = -1;
        yc2 = -1;
        return;
    }
    // clipping candidate
    clipLine(xc11, yc11, xc21, yc21, x1, y1, (x1 + x2) / 2, (y1 + y2) / 2);
    clipLine(xc12, yc12, xc22, yc22, (x1 + x2) / 2, (y1 + y2) / 2, x2, y2);
    if (xc21 == xc12 && yc21 == yc12)
    {
        xc1 = xc11;
        yc1 = yc11;
        xc2 = xc22;
        yc2 = yc22;
    }
    else if (xc11 == -1 && xc21 == -1 && yc11 == -1 && yc21 == -1)
    { // first point invalid
        xc1 = xc12;
        xc2 = xc22;
        yc1 = yc12;
        yc2 = yc22;
    }
    else
    { // second point invalid
        xc1 = xc11;
        xc2 = xc21;
        yc1 = yc11;
        yc2 = yc21;
```

```
    }
}
int main()
{
    initwindow(800, 800);
    int x0, y0, x1, y1;
    cout << "Enter first point: ";
    cin >> x0 >> y0;
    cout << "Enter second point: ";
    cin >> x1 >> y1;
    rectangle(XWMIN, YWMIN, XWMAX, YWMAX);
    line(x0, y0, x1, y1);
    int xc1, yc1, xc2, yc2;
    clipLine(xc1, yc1, xc2, yc2, x0, y0, x1, y1);
    if (xc1 != -1 && yc1 != -1 && xc2 != -1 && yc2 != -1)
    line(xc1, yc1, xc2, yc2);
    cout << xc1 << " " << yc1 << "   " << xc2 << " " << yc2 << endl;
    getch();
    return 0;
}
```

```
tabis@LAPTOP-4D7ISNT3 MINGW64 ~/OneDrive/Desktop/cg_pracs
$ ./prac9.exe
Enter first point: 50 70
Enter second point: 25 70
-1 -1   -1 -1
```