PRACTICAL FILE

of

“Computer Graphics Lab”

PE-CS-A404AL

Submitted To:
Er. Shilpa
Assistant Professor
Deptt. of Computer Science

Submitted By:
Rakshu Chauhan
1220274
B.Tech CSE (8th sem)

Department of Computer Science & Engineering
Seth Jai Parkash Mukand Lal Institute of Engineering & Technology, Radaur – 135133 (Yamuna Nagar)
(Affiliated to Kurukshetra University, Kurukshetra, Haryana, India)

INDEX

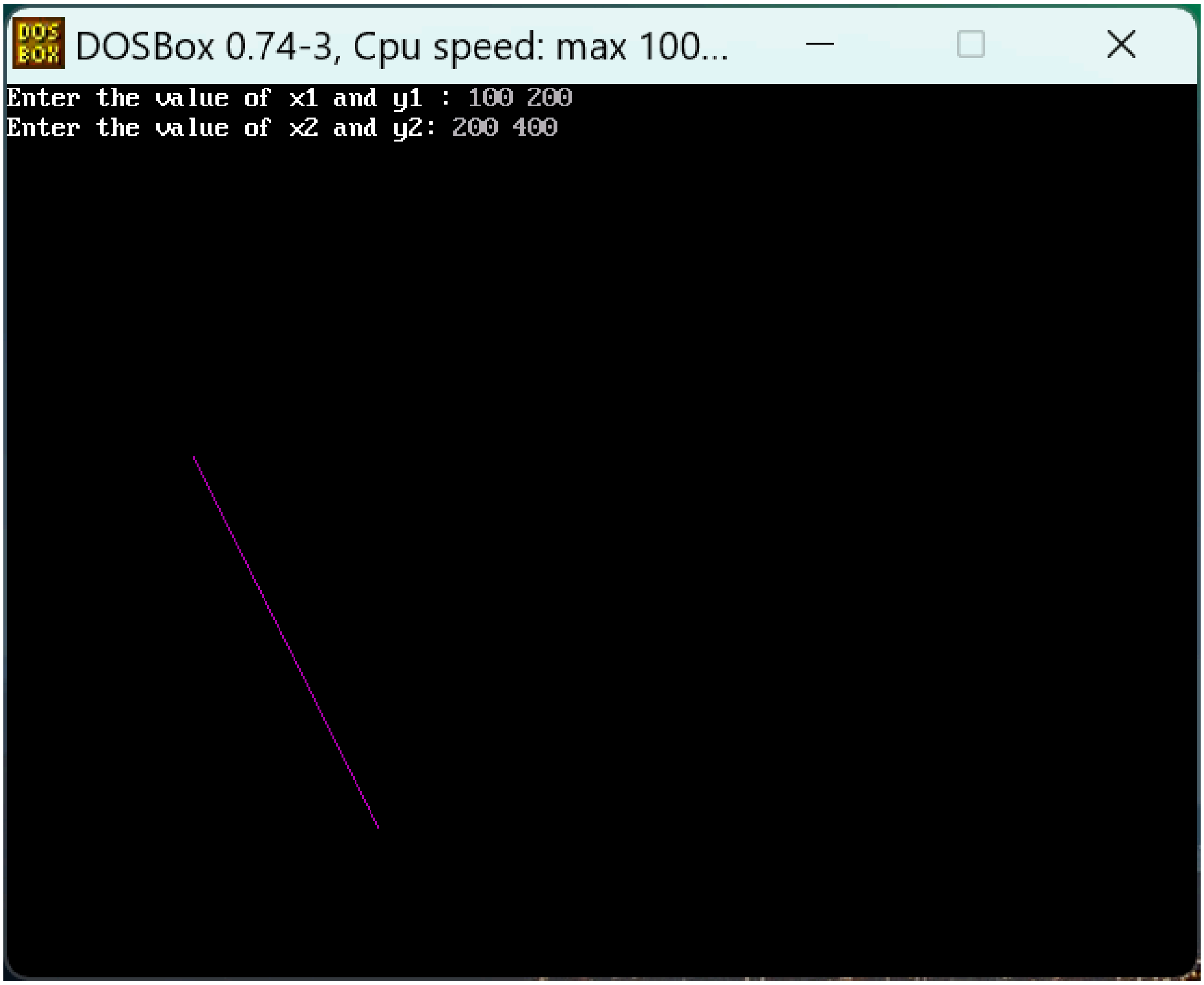
S.No.	Program	Page No.	Date	Remarks
1.	Write a program to implement the DDA line drawing algorithm.	2		
2.	Write a program to implement Bresenham’s line algorithm.	4		
3.	Write a program to implement Bresenham’s circle algorithm.	7		
4.	Write a program to move and object using 2-D transformation.	10		
5.	Write a program to implement the midpoint circle drawing algorithm.	12		
6.	Write a program to implement a line clipping algorithm.	15		
7.	Write a program to implement boundary fill algorithm.	22		
8.	Write a program to implement a polygon clipping algorithm.	24		
9.	Program To implement line clipping using Cohen-Sutherland line clipping algo.	31		
10.	Program To Scale any object.	37		

PRACTICAL NO. – 1

AIM: Write a program to implement the DDA line drawing algorithm.

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

void main()
{
   float x, y, x1, y1, x2, y2, dx, dy, step;
   int i, gd = DETECT, gm;

   initgraph(&gd, &gm, "c:\\turboc3\\bgi");

   cout << "Enter the value of x1 and y1 : ";
   cin >> x1 >> y1;
   cout << "Enter the value of x2 and y2: ";
   cin >> x2 >> y2;

   dx = abs(x2 - x1);
   dy = abs(y2 - y1);

   if (dx >= dy)
      step = dx;
   else
      step = dy;

   dx = dx / step;
   dy = dy / step;

   x = x1;
   y = y1;

   i = 1;
   while (i <= step)
   {
      putpixel(x, y, 5);
      x = x + dx;
      y = y + dy;
      i = i + 1;
      delay(100);
   }
   closegraph();
}
```

Output:



```
Enter the value of x1 and y1 : 100 200
Enter the value of x2 and y2: 200 400
```

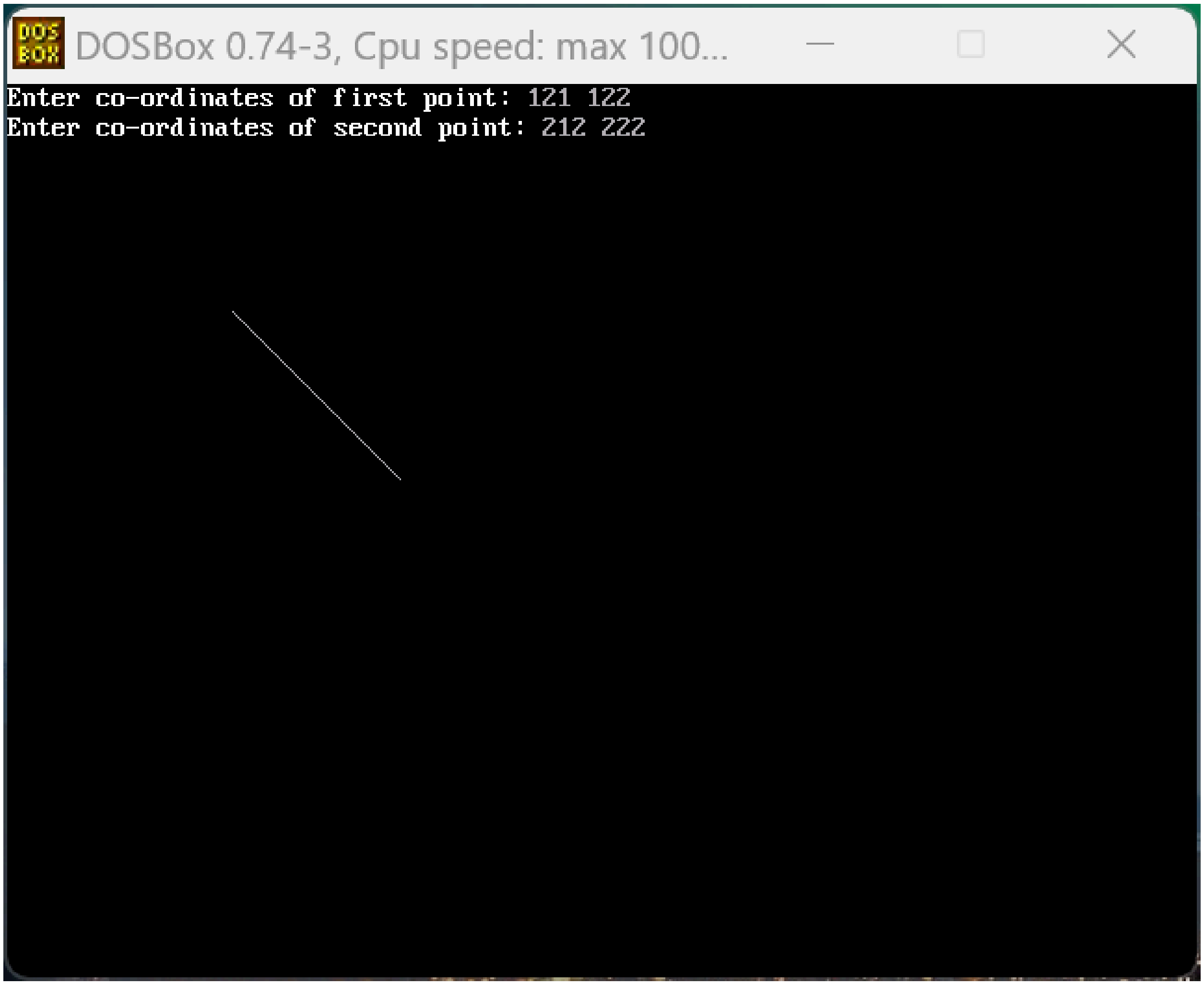
PRACTICAL NO. – 2

AIM: Write a program to implement Bresenham’s line algorithm.

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

void drawline(int x0, int y0, int x1, int y1)
{
   int dx, dy, p, x, y;

   dx = x1 - x0;
   dy = y1 - y0;

   x = x0;
   y = y0;

   p = 2 * dy - dx;

   while (x < x1)
   {
      if (p >= 0)
      {
         putpixel(x, y, 7);
         y = y + 1;
         p = p + 2 * dy - 2 * dx;
      }
      else
      {
         putpixel(x, y, 7);
         p = p + 2 * dy;
      }
      x = x + 1;
   }
}
```

```
int main()
{
   int gdriver = DETECT, gmode, error, x0, y0, x1, y1;
   initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

   cout << "Enter co-ordinates of first point: ";
   cin >> x0 >> y0;

   cout << "Enter co-ordinates of second point: ";
   cin >> x1 >> y1;
   drawline(x0, y0, x1, y1);

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

Output:



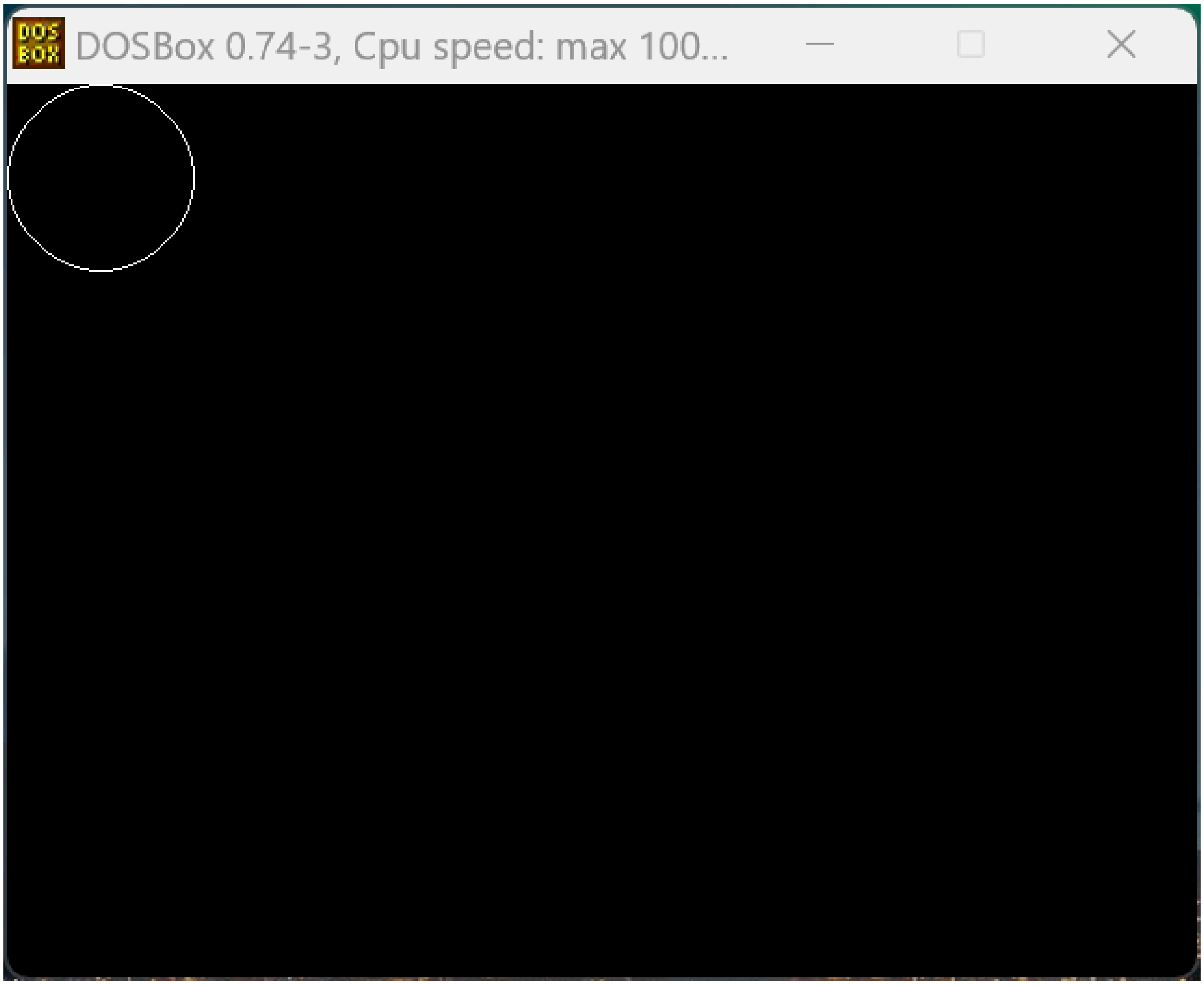
```
Enter co-ordinates of first point: 121 122
Enter co-ordinates of second point: 212 222
```

PRACTICAL NO. – 3

AIM: Write a program to implement Bresenham’s circle algorithm.

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

void drawCircle(int xc, int yc, int x, int y)
{
        putpixel(xc + x, yc + y, WHITE);
        putpixel(xc - x, yc + y, WHITE);
        putpixel(xc + x, yc - y, WHITE);
        putpixel(xc - x, yc - y, WHITE);
        putpixel(xc + y, yc + x, WHITE);
        putpixel(xc - y, yc + x, WHITE);
        putpixel(xc + y, yc - x, WHITE);
        putpixel(xc - y, yc - x, WHITE);
}

// Function for circle-generation  using Bresenham's algorithm
void circleBres(int xc, int yc, int r)
{
        int x = 0, y = r;
        int d = 3 - 2 * r;
        while (y >= x)
        {
                drawCircle(xc, yc, x, y);
                x++;
                if (d > 0)
                {
                        y--;
                        d = d + 4 * (x - y) + 10;
                }
                else
                        d = d + 4 * x + 6;
                drawCircle(xc, yc, x, y);
                delay(50);
        }
}

int main()
{
        int xc = 50, yc = 50, r = 30;
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        circleBres(xc, yc, r);
        getch();
        closegraph();
        return 0;
}
```

Output:

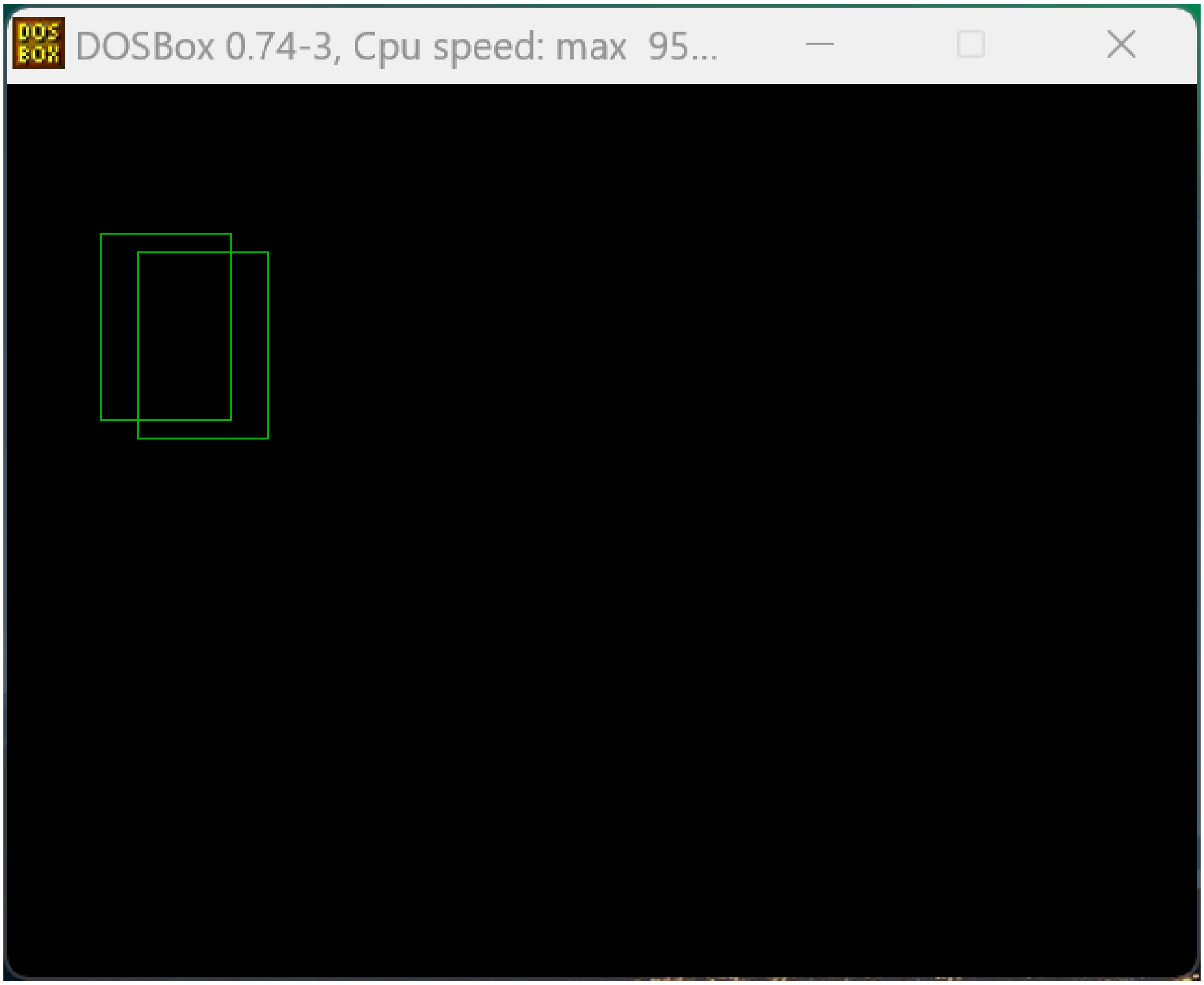


PRACTICAL NO. – 4

AIM: Write a program to move and object using 2-D transformation.

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

// function to translate rectangle
void translateRectangle(int P[][2], int T[])
{

   int gd = DETECT, gm, errorcode;
   initgraph(&gd, &gm, "c:\\turboc3\\bgi");
   setcolor(2);
   // rectangle (Xmin, Ymin, Xmax, Ymax) original rectangle
   rectangle(P[0][0], P[0][1], P[1][0], P[1][1]);

   // calculating translated coordinates
   P[0][0] = P[0][0] + T[0];
   P[0][1] = P[0][1] + T[1];
   P[1][0] = P[1][0] + T[0];
   P[1][1] = P[1][1] + T[1];

   // translated rectangle (Xmin, Ymin, Xmax, Ymax)
   rectangle(P[0][0], P[0][1], P[1][0], P[1][1]);
}

int main()
{
   int P[2][2] = {5, 8, 12, 18};
   int T[] = {2, 1}; // translation factor
   translateRectangle(P, T);
   return 0;
}
```

Output:



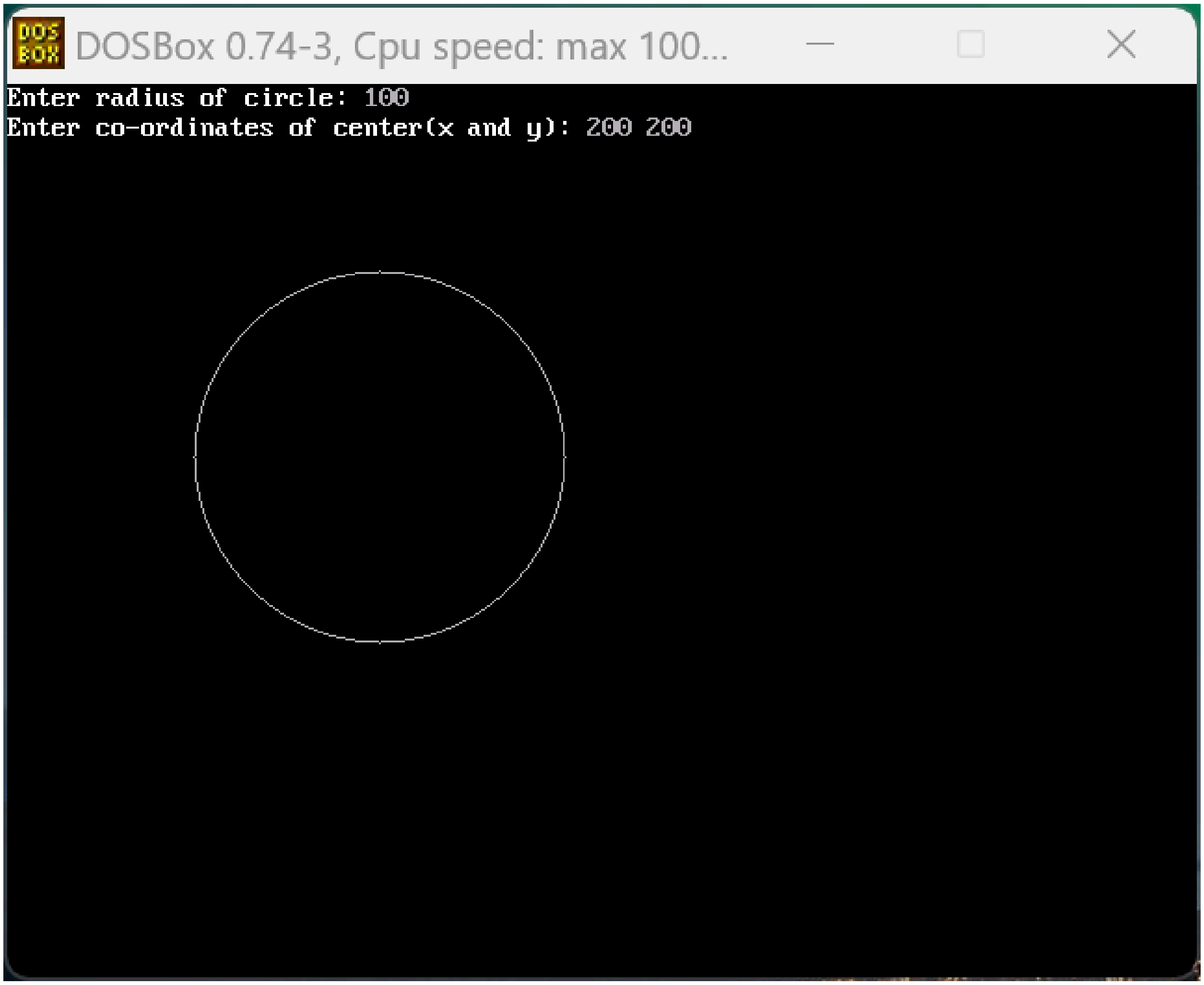
PRACTICAL NO. – 5

AIM: Write a program to implement the midpoint circle drawing algorithm.

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

void drawcircle(int x0, int y0, int radius)
{
   int x = radius;
   int y = 0;
   int err = 0;

   while (x >= y)
   {
      putpixel(x0 + x, y0 + y, 7);
      putpixel(x0 + y, y0 + x, 7);
      putpixel(x0 - y, y0 + x, 7);
      putpixel(x0 - x, y0 + y, 7);
      putpixel(x0 - x, y0 - y, 7);
      putpixel(x0 - y, y0 - x, 7);
      putpixel(x0 + y, y0 - x, 7);
      putpixel(x0 + x, y0 - y, 7);

      if (err <= 0)
      {
         y += 1;
         err += 2 * y + 1;
      }

      if (err > 0)
      {
         x -= 1;
         err -= 2 * x + 1;
      }
   }
}
```

```
int main()
{
   int gdriver = DETECT, gmode, error, x, y, r;
   initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

   cout << "Enter radius of circle: ";
   cin >> r;

   cout << "Enter co-ordinates of center(x and y): ";
   cin >> x >> y;
   drawcircle(x, y, r);
   getch();
   closegraph();
   return 0;
}
```

Output:



```
Enter radius of circle: 100
Enter co-ordinates of center(x and y): 200 200
```

PRACTICAL NO. – 6

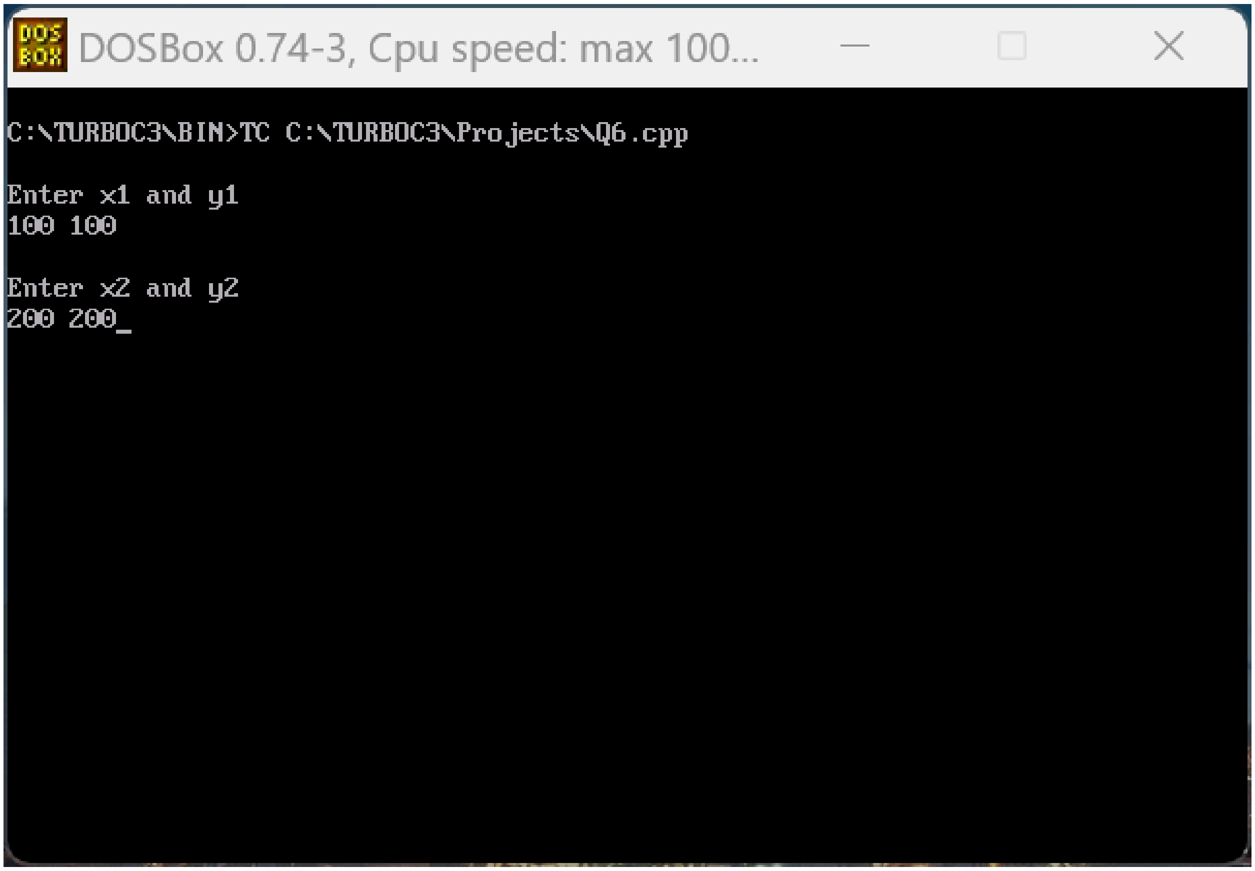
AIM: Write a program to implement a line clipping algorithm.

```
#include <iostream.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>
#include <dos.h>

typedef struct coordinate
{
   int x, y;
   char code[4];
} PT;

void drawwindow();
void drawline(PT p1, PT p2);
PT setcode(PT p);
int visibility(PT p1, PT p2);
PT resetendpt(PT p1, PT p2);

void main()
{
   int gd = DETECT, v, gm;
   PT p1, p2, p3, p4, ptemp;

   cout << "\nEnter x1 and y1\n";
   cin >> p1.x >> p1.y;
   cout << "\nEnter x2 and y2\n";
   cin >> p2.x >> p2.y;

   initgraph(&gd, &gm, "c:\\turboc3\\bgi");
   drawwindow();
   delay(500);

   drawline(p1, p2);
   delay(500);
   cleardevice();
```

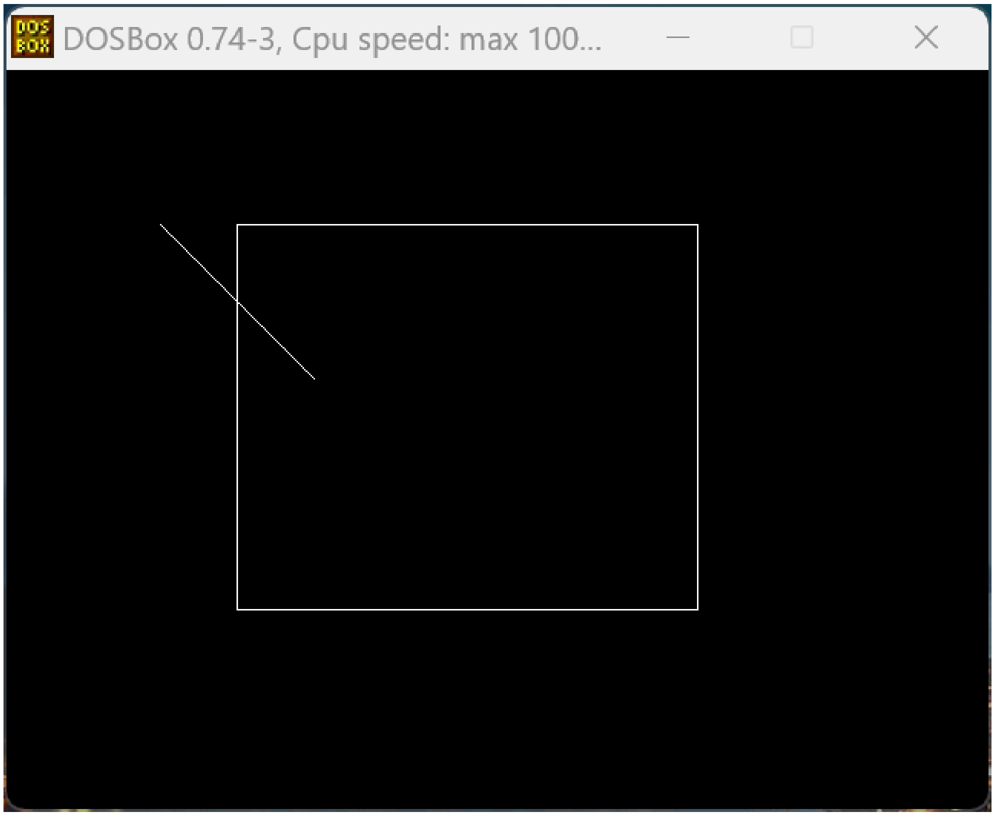
```
    delay(500);
    p1 = setcode(p1);
    p2 = setcode(p2);
    v = visibility(p1, p2);
    delay(500);

    switch (v)
    {
    case 0:
        drawwindow();
        delay(500);
        drawline(p1, p2);
        break;
    case 1:
        drawwindow();
        delay(500);
        break;
    case 2:
        p3 = resetendpt(p1, p2);
        p4 = resetendpt(p2, p1);
        drawwindow();
        delay(500);
        drawline(p3, p4);
        break;
    }

    delay(500);
    getch();
    closegraph();
}

void drawwindow()
{
    line(150, 100, 450, 100);
    line(450, 100, 450, 350);
    line(450, 350, 150, 350);
    line(150, 350, 150, 100);
}

void drawline(PT p1, PT p2)
```

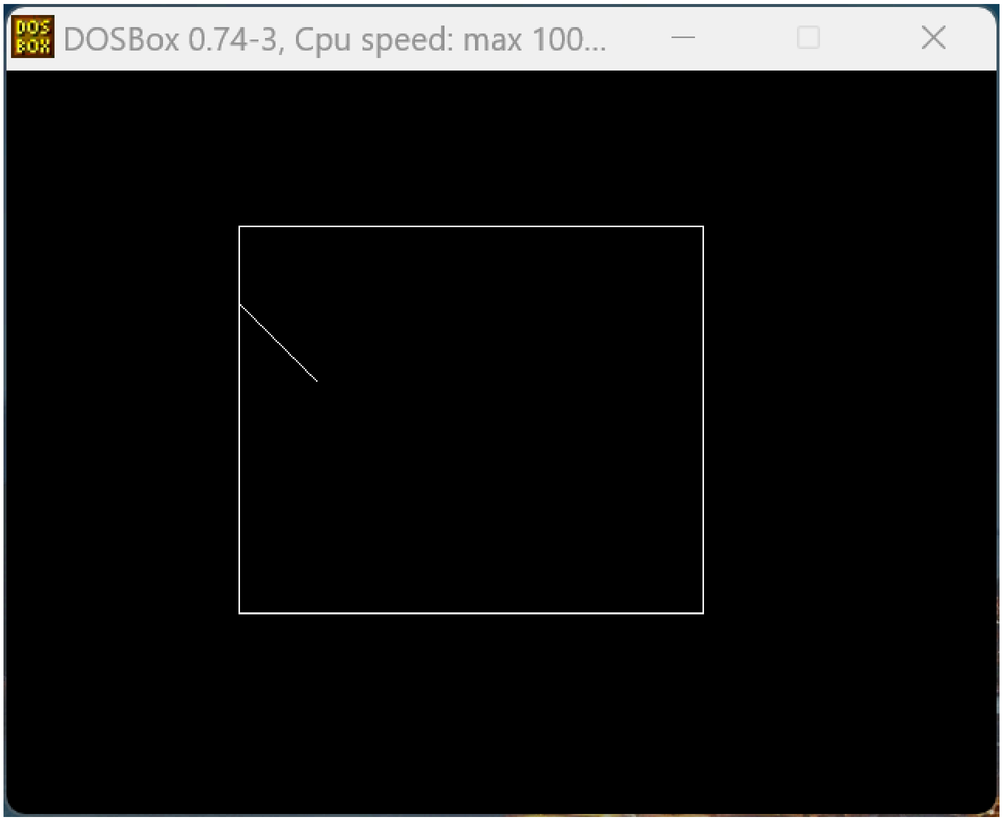
```
{
   line(p1.x, p1.y, p2.x, p2.y);
}

PT setcode(PT p) // for setting the 4 bit code
{
   PT ptemp;

   if (p.y < 100)
      ptemp.code[0] = '1'; // Top
   else
      ptemp.code[0] = '0';

   if (p.y > 350)
      ptemp.code[1] = '1'; // Bottom
   else
      ptemp.code[1] = '0';

   if (p.x > 450)
      ptemp.code[2] = '1'; // Right
   else
      ptemp.code[2] = '0';

   if (p.x < 150)
      ptemp.code[3] = '1'; // Left
   else
      ptemp.code[3] = '0';

   ptemp.x = p.x;
   ptemp.y = p.y;

   return (ptemp);
}

int visibility(PT p1, PT p2)
{
   int i, flag = 0;

   for (i = 0; i < 4; i++)
   {
        if ((p1.code[i] != '0') || (p2.code[i] != '0'))
            flag = 1;
    }

    if (flag == 0)
        return (0);

    for (i = 0; i < 4; i++)
    {
        if ((p1.code[i] == p2.code[i]) && (p1.code[i] == '1'))
            flag = '0';
    }

    if (flag == 0)
        return (1);

    return (2);
}

PT resetendpt(PT p1, PT p2)
{
    PT temp;
    int x, y, i;
    float m, k;

    if (p1.code[3] == '1')
        x = 150;

    if (p1.code[2] == '1')
        x = 450;

    if ((p1.code[3] == '1') || (p1.code[2] == '1'))
    {
        m = (float)(p2.y - p1.y) / (p2.x - p1.x);
        k = (p1.y + (m * (x - p1.x)));
        temp.y = k;
        temp.x = x;

        for (i = 0; i < 4; i++)
            temp.code[i] = p1.code[i];
        if (temp.y <= 350 && temp.y >= 100)
            return (temp);
    }

    if (p1.code[0] == '1')
        y = 100;

    if (p1.code[1] == '1')
        y = 350;

    if ((p1.code[0] == '1') || (p1.code[1] == '1'))
    {
        m = (float)(p2.y - p1.y) / (p2.x - p1.x);
        k = (float)p1.x + (float)(y - p1.y) / m;
        temp.x = k;
        temp.y = y;

        for (i = 0; i < 4; i++)
            temp.code[i] = p1.code[i];

        return (temp);
    }
    else
        return (p1);
}
```

Output:



```
C:\TURBOC3\BIN>TC C:\TURBOC3\Projects\Q6.cpp

Enter x1 and y1
100 100

Enter x2 and y2
200 200_
```


DOSBox 0.74-3, Cpu speed: max 100...

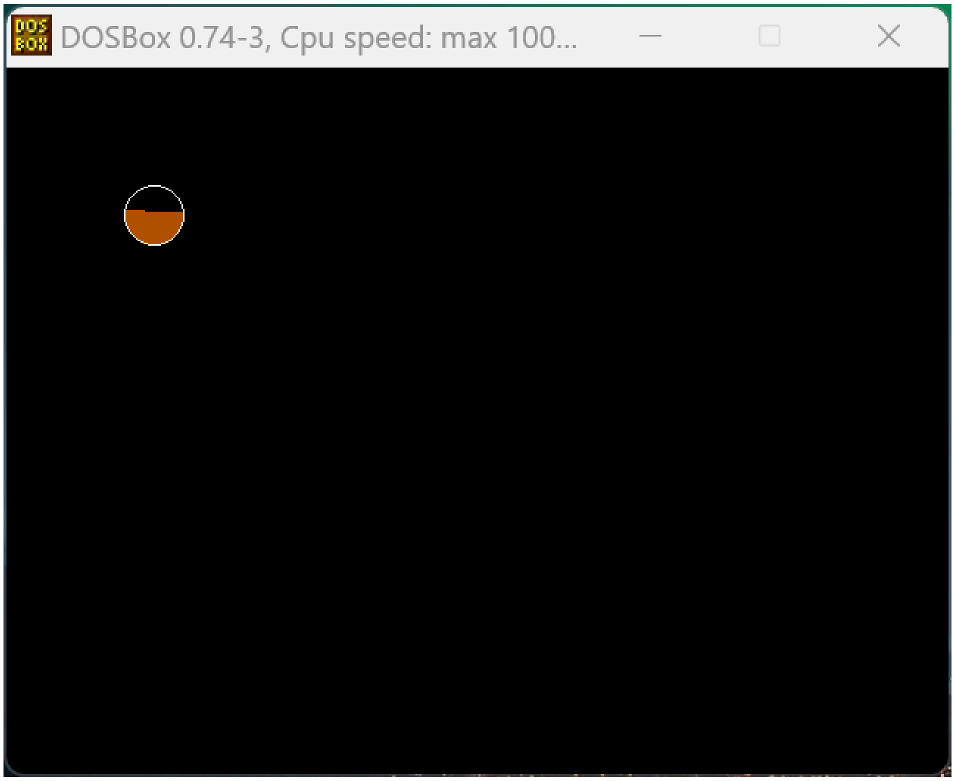

DOSBox 0.74-3, Cpu speed: max 100...

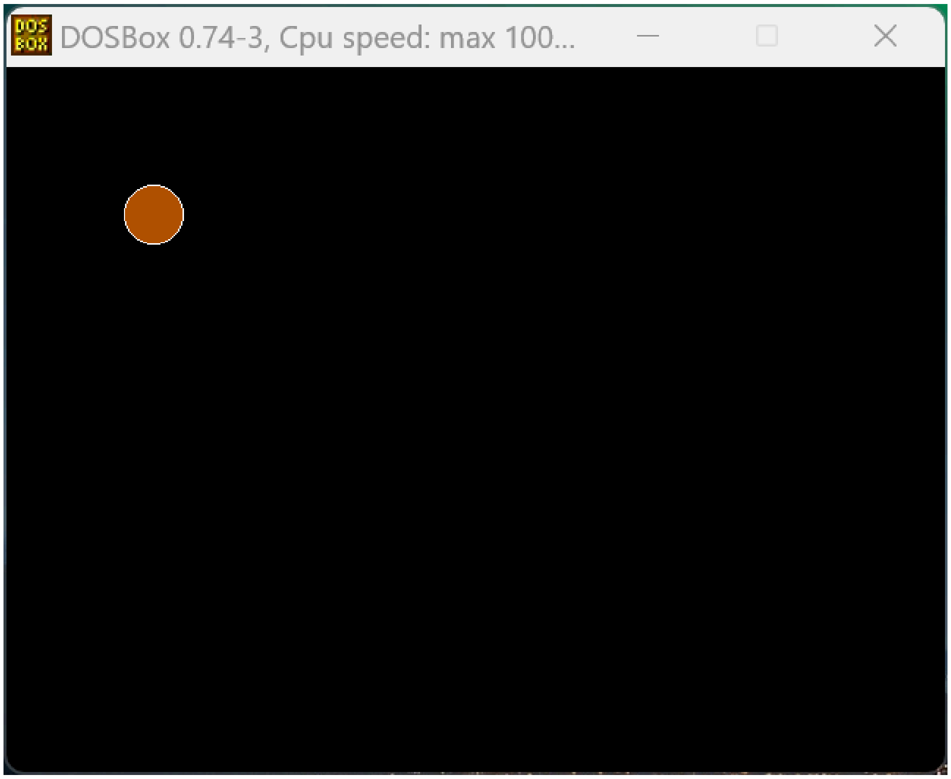
PRACTICAL NO. – 7

AIM: Write a program to implement boundary fill algorithm.

```
#include <graphics.h>
#include <conio.h>
#include<dos.h>
void boundaryFill4(int x, int y, int fill_color, int boundary_color)
{
   if (getpixel(x, y) != boundary_color && getpixel(x, y) != fill_color)
   {
      putpixel(x, y, fill_color);
      delay(10);
      boundaryFill4(x + 1, y, fill_color, boundary_color);
      boundaryFill4(x-1, y, fill_color, boundary_color);
      boundaryFill4(x, y+1, fill_color, boundary_color);
      boundaryFill4(x, y - 1, fill_color, boundary_color);
   }
}

int main()
{
   // gm is Graphics mode which is  a computer display mode that generates image using pixels.
   // DETECT is a macro defined in "graphics.h" header file
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "c:\\turboc3\\bgi");

   int x = 100, y = 100, radius = 50;

   circle(x, y, radius);
   boundaryFill4(x, y, 6, 15);

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

Output:





PRACTICAL NO. – 8

AIM: Write a program to implement a polygon clipping algorithm.

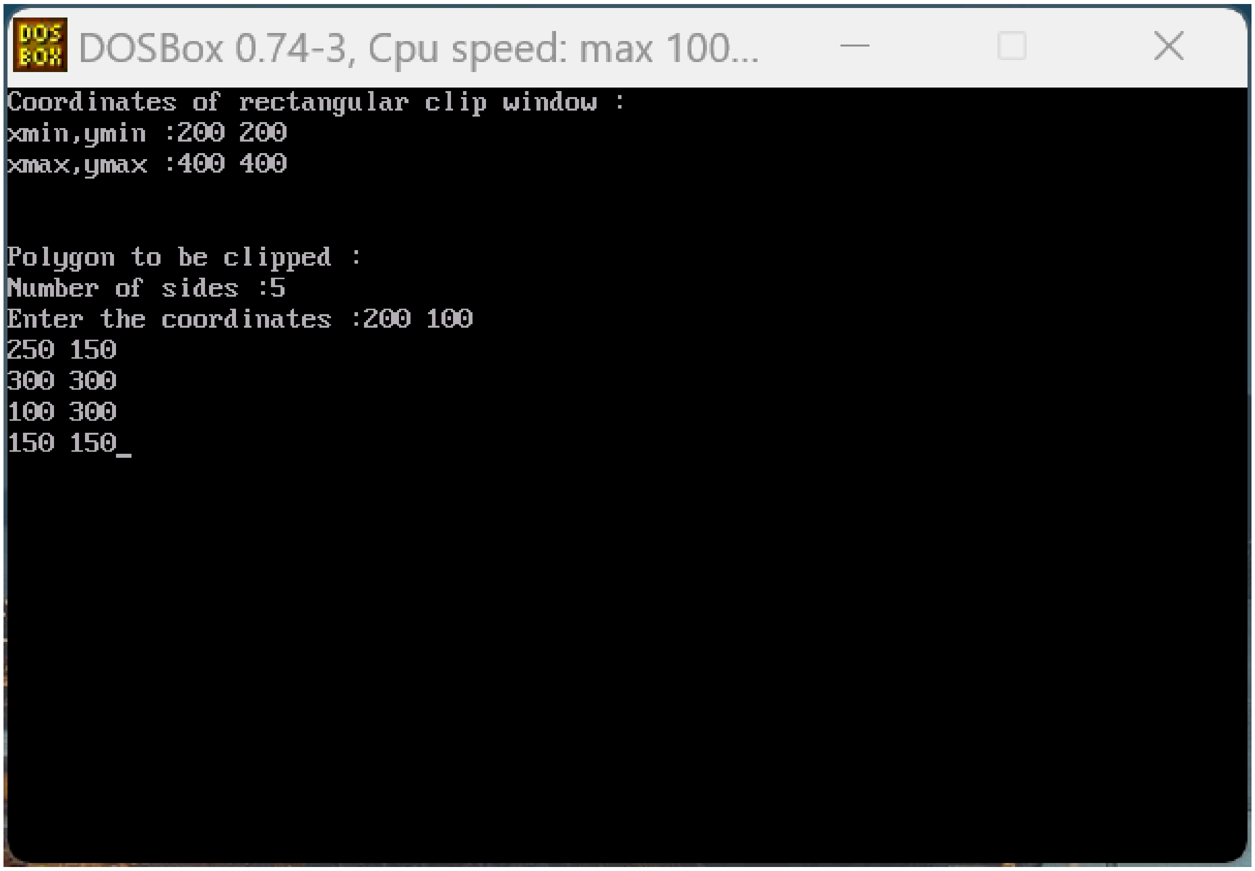
```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
#define round(a) ((int)(a + 0.5))
int k;
float xmin, ymin, xmax, ymax, arr[20], m;
void clipl(float x1, float y1, float x2, float y2)
{
   if (x2 - x1)
      m = (y2 - y1) / (x2 - x1);
   else
      m = 100000;
   if (x1 >= xmin && x2 >= xmin)
   {
      arr[k] = x2;
      arr[k + 1] = y2;
      k += 2;
   }
   if (x1 < xmin && x2 >= xmin)
   {
      arr[k] = xmin;
      arr[k + 1] = y1 + m * (xmin - x1);
      arr[k + 2] = x2;
      arr[k + 3] = y2;
      k += 4;
   }
   if (x1 >= xmin && x2 < xmin)
   {
      arr[k] = xmin;
      arr[k + 1] = y1 + m * (xmin - x1);
      k += 2;
   }
}
void clipt(float x1, float y1, float x2, float y2)
{
   if (y2 - y1)
      m = (x2 - x1) / (y2 - y1);
```

```
    else
        m = 100000;
    if (y1 <= ymax && y2 <= ymax)
    {
        arr[k] = x2;
        arr[k + 1] = y2;
        k += 2;
    }
    if (y1 > ymax && y2 <= ymax)
    {
        arr[k] = x1 + m * (ymax - y1);
        arr[k + 1] = ymax;
        arr[k + 2] = x2;
        arr[k + 3] = y2;
        k += 4;
    }
    if (y1 <= ymax && y2 > ymax)
    {
        arr[k] = x1 + m * (ymax - y1);
        arr[k + 1] = ymax;
        k += 2;
    }
}
void clipr(float x1, float y1, float x2, float y2)
{
    if (x2 - x1)
        m = (y2 - y1) / (x2 - x1);
    else
        m = 100000;
    if (x1 <= xmax && x2 <= xmax)
    {
        arr[k] = x2;
        arr[k + 1] = y2;
        k += 2;
    }
    if (x1 > xmax && x2 <= xmax)
    {
        arr[k] = xmax;
        arr[k + 1] = y1 + m * (xmax - x1);
        arr[k + 2] = x2;
        arr[k + 3] = y2;
        k += 4;
    }
    if (x1 <= xmax && x2 > xmax)
    {
        arr[k] = xmax;
        arr[k + 1] = y1 + m * (xmax - x1);
        k += 2;
    }
}
void clipb(float x1, float y1, float x2, float y2)
{
    if (y2 - y1)
        m = (x2 - x1) / (y2 - y1);
    else
        m = 100000;
    if (y1 >= ymin && y2 >= ymin)
    {
        arr[k] = x2;
        arr[k + 1] = y2;
        k += 2;
    }
    if (y1 < ymin && y2 >= ymin)
    {
        arr[k] = x1 + m * (ymin - y1);
        arr[k + 1] = ymin;
        arr[k + 2] = x2;

        arr[k + 3] = y2;
        k += 4;
    }
    if (y1 >= ymin && y2 < ymin)
    {
        arr[k] = x1 + m * (ymin - y1);
        arr[k + 1] = ymin;
        k += 2;
    }
}
void main()
{
```

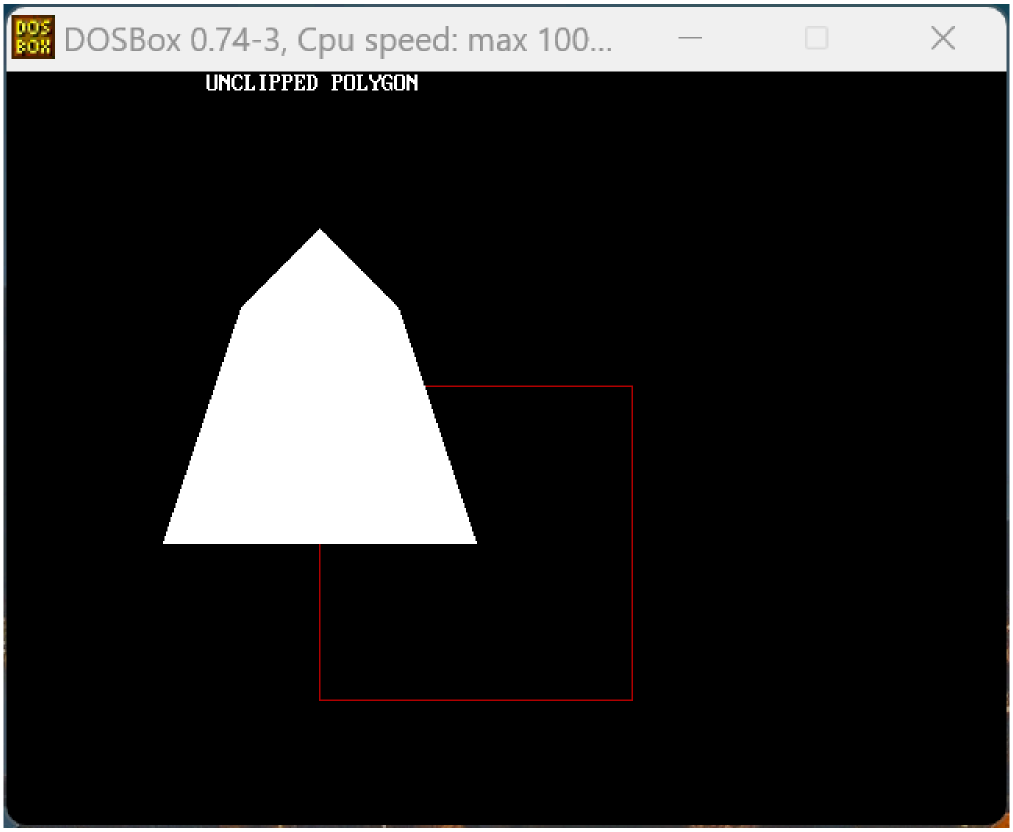
```
int gd = DETECT, gm, n, poly[20];
float xi, yi, xf, yf, polyy[20];
clrscr();
cout << "Coordinates of rectangular clip window :\nxmin,ymin :";
cin >> xmin >> ymin;
cout << "xmax,ymax :";
cin >> xmax >> ymax;
cout << "\n\nPolygon to be clipped :\nNumber of sides :";
cin >> n;
cout << "Enter the coordinates :";
for (int i = 0; i < 2 * n; i++)
   cin >> polyy[i];
polyy[i] = polyy[0];
polyy[i + 1] = polyy[1];
for (i = 0; i < 2 * n + 2; i++)
   poly[i] = round(polyy[i]);

initgraph(&gd, &gm, "c:\\turboc3\\bgi");
setcolor(RED);
rectangle(xmin, ymax, xmax, ymin);
cout << "\t\tUNCLIPPED POLYGON";
setcolor(WHITE);
fillpoly(n, poly);
getch();
cleardevice();
k = 0;
for (i = 0; i < 2 * n; i += 2)

   clipl(polyy[i], polyy[i + 1], polyy[i + 2], polyy[i + 3]);

n = k / 2;
for (i = 0; i < k; i++)

   polyy[i] = arr[i];
polyy[i] = polyy[0];
polyy[i + 1] = polyy[1];
k = 0;
for (i = 0; i < 2 * n; i += 2)

   clipt(polyy[i], polyy[i + 1], polyy[i + 2], polyy[i + 3]);
```

```
    n = k / 2;
    for (i = 0; i < k; i++)

        polyy[i] = arr[i];
    polyy[i] = polyy[0];
    polyy[i + 1] = polyy[1];
    k = 0;

    for (i = 0; i < 2 * n; i += 2)

        clipr(polyy[i], polyy[i + 1], polyy[i + 2], polyy[i + 3]);

    n = k / 2;
    for (i = 0; i < k; i++)

        polyy[i] = arr[i];
    polyy[i] = polyy[0];
    polyy[i + 1] = polyy[1];
    k = 0;
    for (i = 0; i < 2 * n; i += 2)

        clipb(polyy[i], polyy[i + 1], polyy[i + 2], polyy[i + 3]);

    for (i = 0; i < k; i++)

        poly[i] = round(arr[i]);

    if (k)

        fillpoly(k / 2, poly);

    setcolor(RED);
    rectangle(xmin, ymax, xmax, ymin);
    cout << "\tCLIPPED POLYGON";
    getch();
    closegraph();
}
```

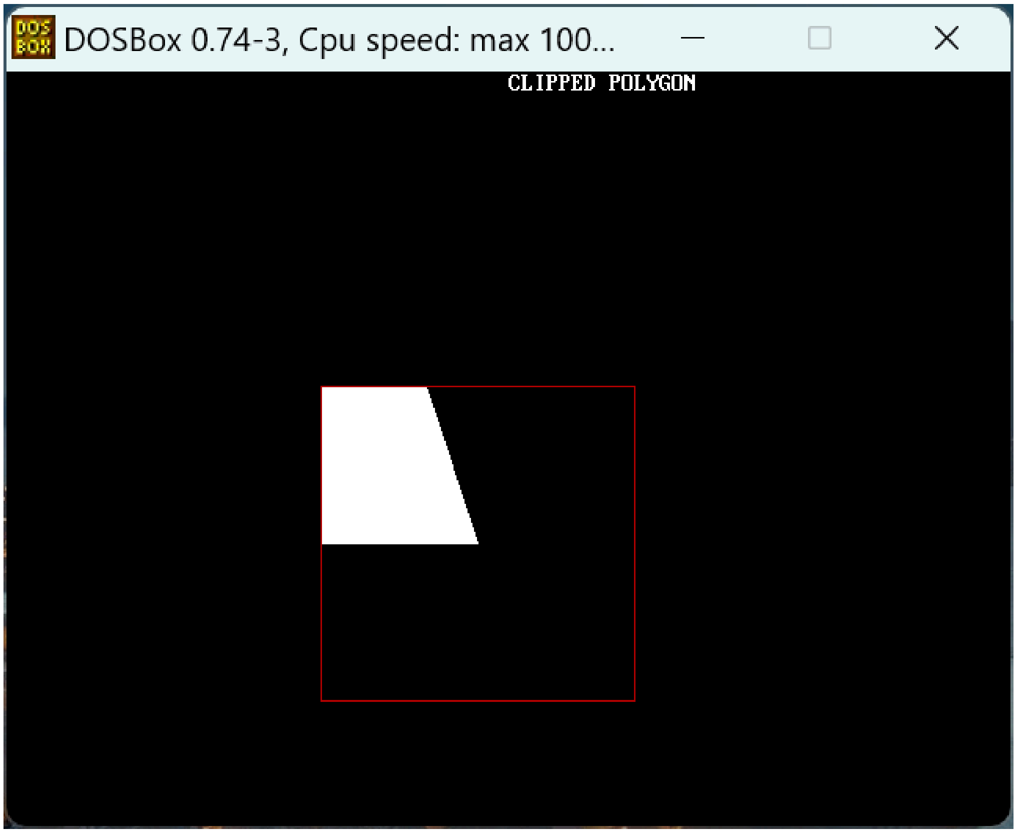
Output:



```
Coordinates of rectangular clip window :
xmin,ymin :200 200
xmax,ymax :400 400


Polygon to be clipped :
Number of sides :5
Enter the coordinates :200 100
250 150
300 300
100 300
150 150_
```



UNCLIPPED POLYGON



CLIPPED POLYGON

PRACTICAL NO. – 9

AIM: Program To implement line clipping using Cohen-Sutherland line clipping algo.

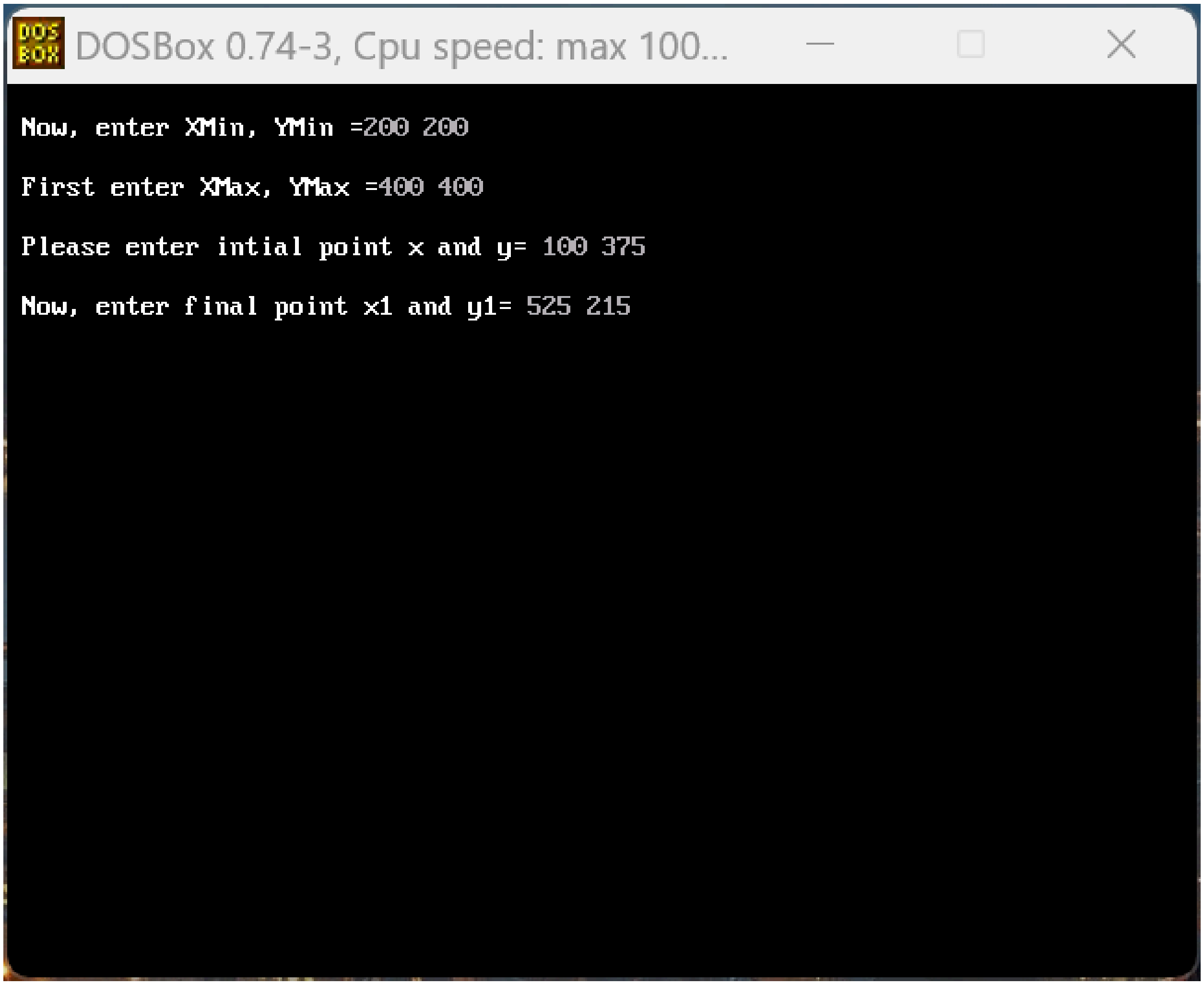
```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <math.h>
void main()
{
  int rcode_begin[4] = {0, 0, 0, 0}, rcode_end[4] = {0, 0, 0, 0}, region_code[4];
  int W_xmax, W_ymax, W_xmin, W_ymin, flag = 0;
  float slope;
  int x, y, x1, y1, i, xc, yc;
  int gr = DETECT, gm;
  initgraph(&gr, &gm, "c:\\turboc3\\bgi");
  printf("\n Now, enter XMin, YMin =");

  scanf("%d %d", &W_xmin, &W_ymin);
  printf("\n First enter XMax, YMax =");
  scanf("%d %d", &W_xmax, &W_ymax);
  printf("\n Please enter intial point x and y= ");
  scanf("%d %d", &x, &y);
  printf("\n Now, enter final point x1 and y1= ");
  scanf("%d %d", &x1, &y1);
  cleardevice();
  rectangle(W_xmin, W_ymin, W_xmax, W_ymax);
  line(x, y, x1, y1);
  line(0, 0, 600, 0);
  line(0, 0, 0, 600);
  if (y > W_ymax)
  {
    rcode_begin[0] = 1;
    flag = 1;
  }
  if (y < W_ymin)
  {
    rcode_begin[1] = 1;
    flag = 1;
```

```
}
if (x > W_xmax)
{
    rcode_begin[2] = 1;
    flag = 1;
}
if (x < W_xmin)
{
    rcode_begin[3] = 1;
    flag = 1;
}
if (y1 > W_ymax)
{
    rcode_end[0] = 1;
    flag = 1;
}
if (y1 < W_ymin)
{
    rcode_end[1] = 1;
    flag = 1;
}
if (x1 > W_xmax)
{
    rcode_end[2] = 1;
    flag = 1;
}
if (x1 < W_xmin)
{
    rcode_end[3] = 1;
    flag = 1;
}
if (flag == 0)
{
    printf("No need of clipping as it is already in window");
}
flag = 1;
for (i = 0; i < 4; i++)
{
    region_code[i] = rcode_begin[i] && rcode_end[i];
    if (region_code[i] == 1)
```

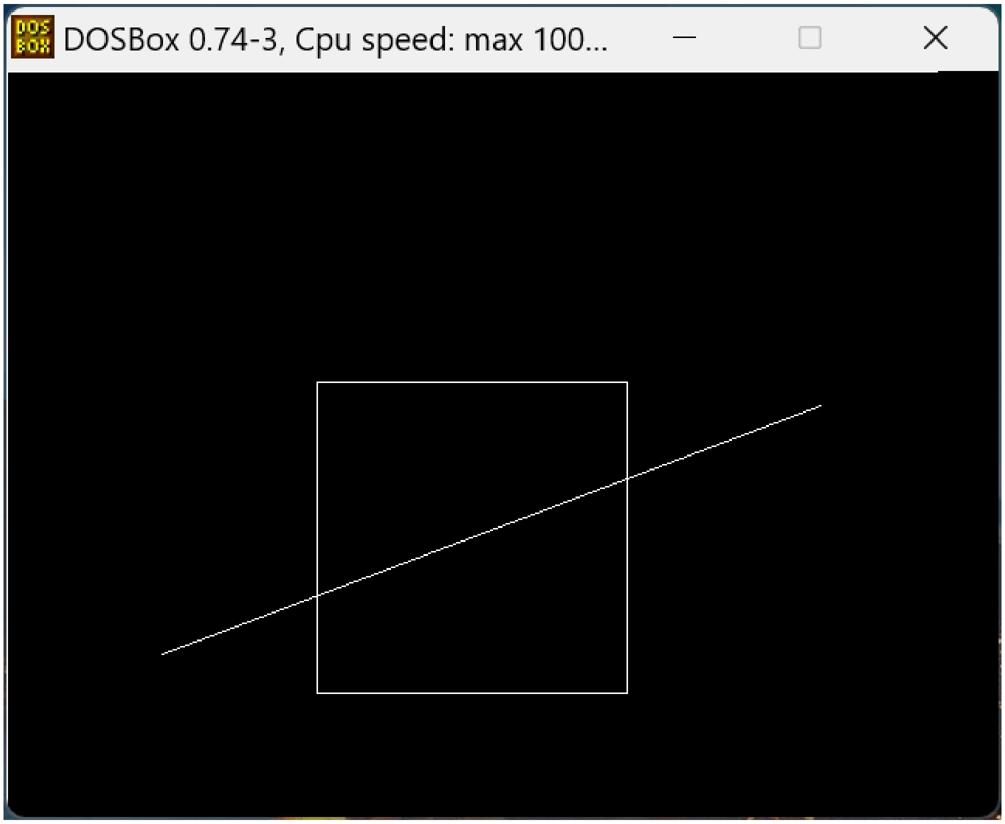
```
        flag = 0;
}
if (flag == 0)
{
    printf("\n Line is completely outside the window");
}
else
{
    slope = (float)(y1 - y) / (x1 - x);
    if (rcode_begin[2] == 0 && rcode_begin[3] == 1) // left
    {
        y = y + (float)(W_xmin - x) * slope;
        x = W_xmin;
    }
    if (rcode_begin[2] == 1 && rcode_begin[3] == 0) // right
    {
        y = y + (float)(W_xmax - x) * slope;
        x = W_xmax;
    }
    if (rcode_begin[0] == 1 && rcode_begin[1] == 0) // top
    {
        x = x + (float)(W_ymax - y) / slope;
        y = W_ymax;
    }
    if (rcode_begin[0] == 0 && rcode_begin[1] == 1)
    {
        x = x + (float)(W_ymin - y) / slope;
        y = W_ymin;
    }
    if (rcode_end[2] == 0 && rcode_end[3] == 1)
    {
        y1 = y1 + (float)(W_xmin - x1) * slope;
        x1 = W_xmin;
    }
    if (rcode_end[2] == 1 && rcode_end[3] == 0)
    {
        y1 = y1 + (float)(W_xmax - x1) * slope;
        x1 = W_xmax;
    }
    if (rcode_end[0] == 1 && rcode_end[1] == 0)
```

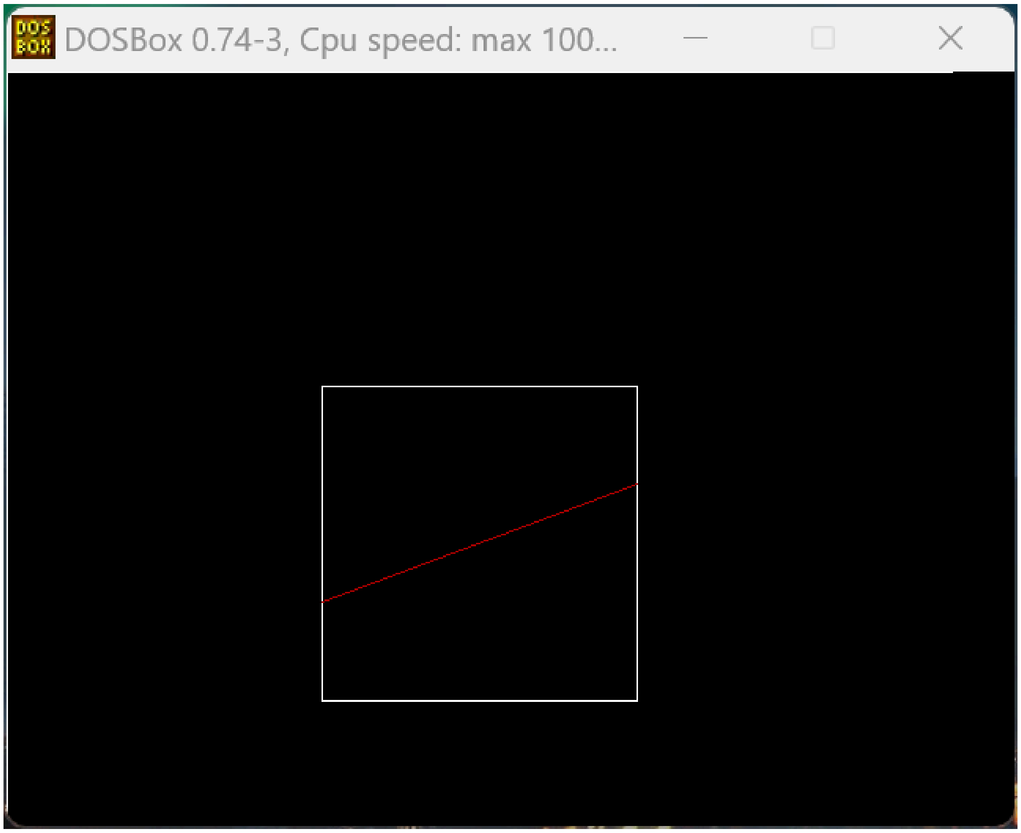
```
        {
          x1 = x1 + (float)(W_ymax - y1) / slope;
          y1 = W_ymax;
        }
        if (rcode_end[0] == 0 && rcode_end[1] == 1)
        {
          x1 = x1 + (float)(W_ymin - y1) / slope;
          y1 = W_ymin;
        }
      }
      delay(1000);
      clearviewport();
      rectangle(W_xmin, W_ymin, W_xmax, W_ymax);
      line(0, 0, 600, 0);
      line(0, 0, 0, 600);
      setcolor(RED);
      line(x, y, x1, y1);
      getch();
      closegraph();
    }
```

Output:



```
Now, enter XMin, YMin =200 200

First enter XMax, YMax =400 400

Please enter intial point x and y= 100 375

Now, enter final point x1 and y1= 525 215
```


DOSBox 0.74-3, Cpu speed: max 100...

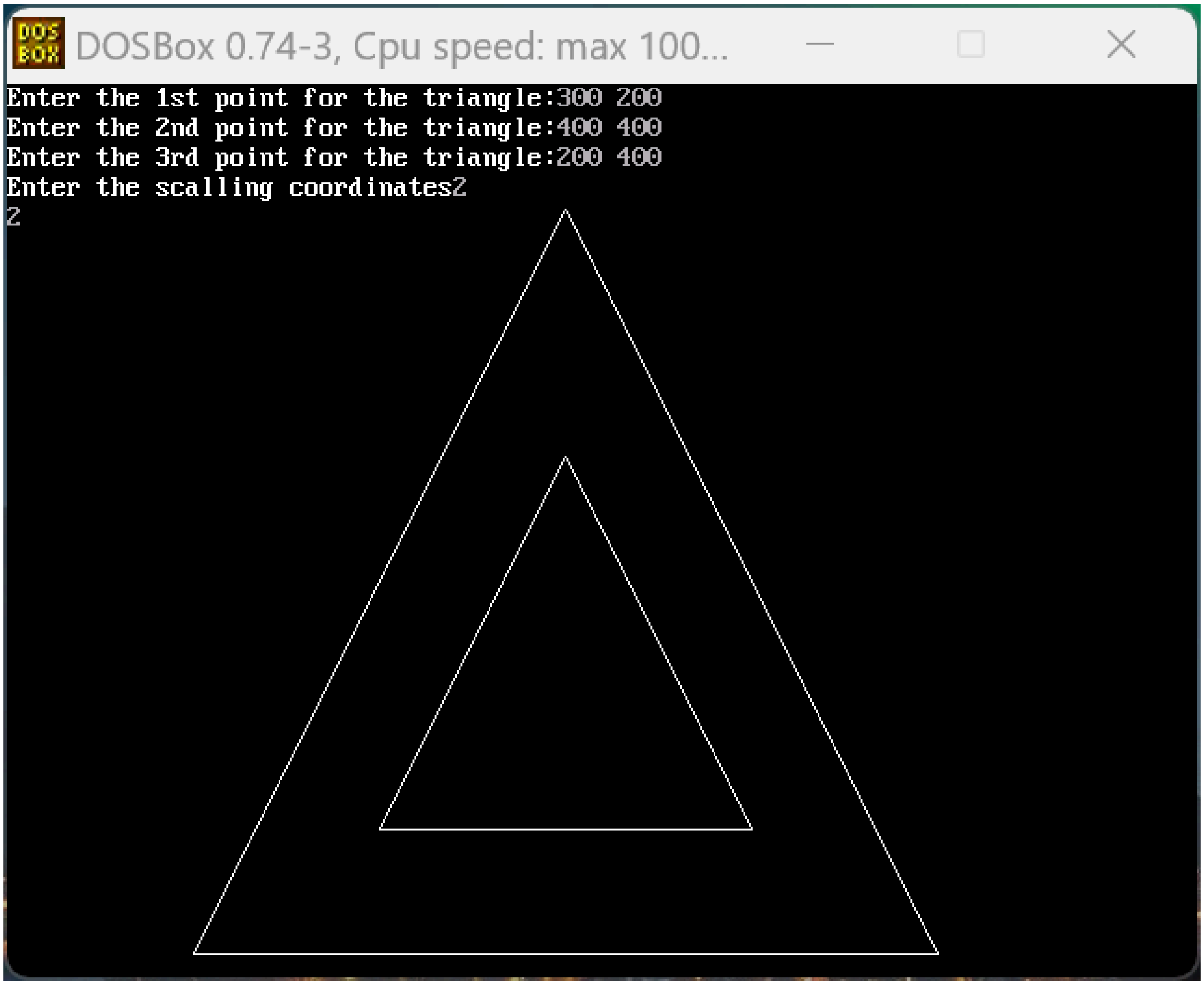

DOSBox 0.74-3, Cpu speed: max 100...

PRACTICAL NO. – 10

AIM: Program To Scale any object.

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
int x1, y1, x2, y2, x3, y3, mx, my;
void main()
{
   int gd = DETECT, gm;
   int x, y, a1, a2, a3, b1, b2, b3;
   int mx, my;
   int c;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   printf("Enter the 1st point for the triangle:");
   scanf("%d%d", &x1, &y1);
   printf("Enter the 2nd point for the triangle:");
   scanf("%d%d", &x2, &y2);
   printf("Enter the 3rd point for the triangle:");
   scanf("%d%d", &x3, &y3);
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);
   printf("Enter the scalling coordinates");
   scanf("%d%d", &x, &y);
   mx = (x1 + x2 + x3) / 3;
   my = (y1 + y2 + y3) / 3;
   a1 = mx + (x1 - mx) * x;
   b1 = my + (y1 - my) * y;
   a2 = mx + (x2 - mx) * x;
   b2 = my + (y2 - my) * y;
   a3 = mx + (x3 - mx) * x;
   b3 = my + (y3 - my) * y;
   line(a1, b1, a2, b2);
   line(a2, b2, a3, b3);
   line(a3, b3, a1, b1);
   getch();
}
```

Output:



```
Enter the 1st point for the triangle:300 200
Enter the 2nd point for the triangle:400 400
Enter the 3rd point for the triangle:200 400
Enter the scalling coordinates2
2
```