Experiment 1

AIM:- To implement DDA Line Drawing Algorithm in C.

ALGORITHM:-

1. Get the input of the two end point (Xo, Yo) AND (X1,Y1).

2. Calculate the difference between the two end points

dx = x0 - yo

dy = X1-y1

3. Based on the calculation difference in Step 2 you need to identify the number

of steps to putpixel if dx>dy, then you need more steps in x co-ordinate, otherwise in y co-ordinate.

```
if (absolute(dx)>absolute(dy))
steps = absolute(dx);
else
steps = absolute(dy);
```

4. Calculate the x increment in x co-ordinate and y co-ordinate

```
Xincrement =dx/(float)steps;
Yincrement = dy/(float)steps;
```

5. Put the pixel by successfully incrementing x and y co-ordinate accordingly

and complete the drawing of the line.

```
for (int v=0; v<Steps;v++)
{
```

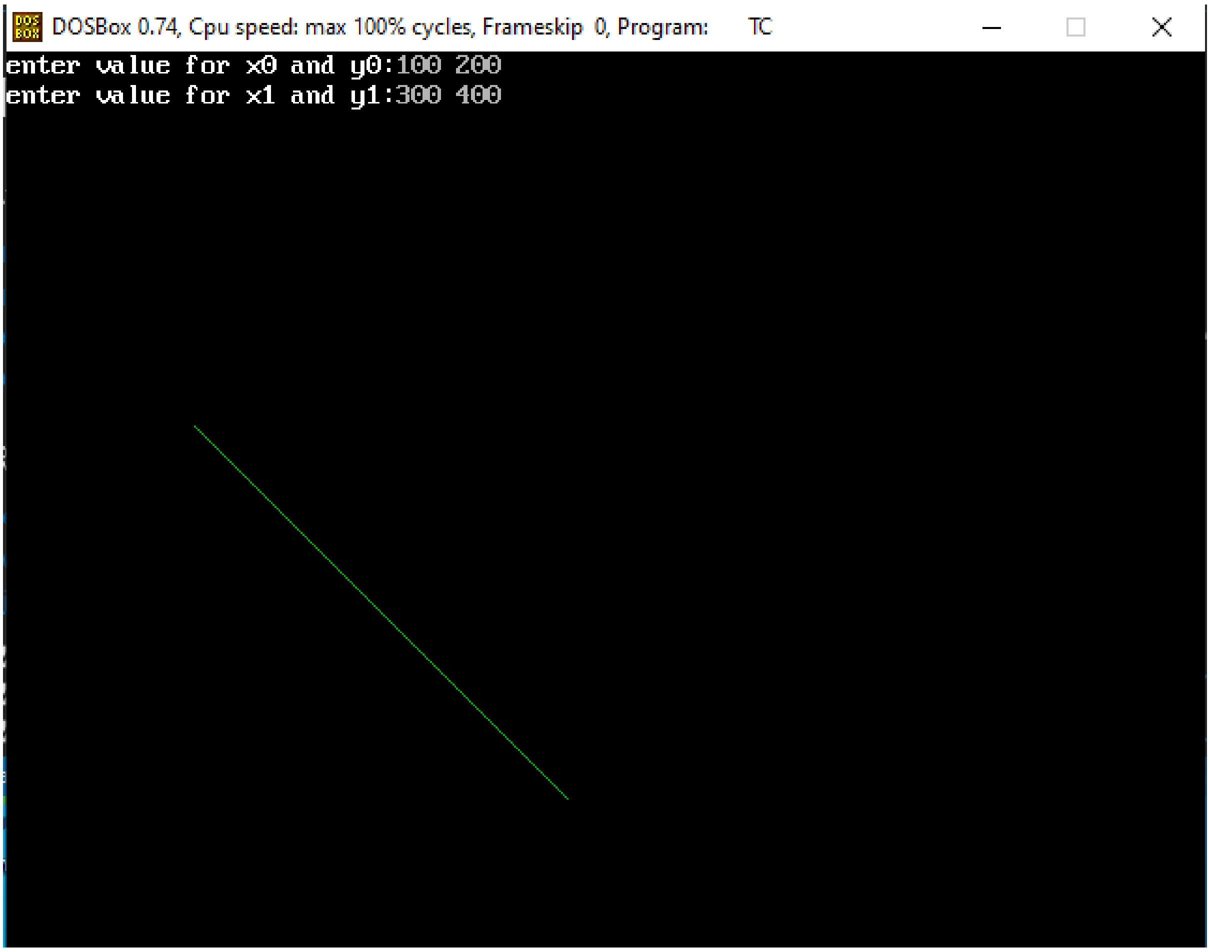
```
x=x+ Xincrement;
y=y+ Yincrement;
putpixel

    }
```

CODE:-

```
#include<stdio.h>
#include <conio.h>
#include <graphics.h>
void main()
{
int gd=DETECT,gm, i;
float x,y,dx,dy,step;
int x1,x0,y1,0; initgraph(&gd, &gm, "C\\TURBOC3\\BGI”);

dx=(x1-x0);
dy=(y1-y0);
if (dx>=dy)
{
step=dx;
}
Else
{
step=dy;
}
dx=dx/step; dy=dy/step;
x=x0);
y=y0;
i=1;
```

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

OUTPUT:-



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
enter value for x0 and y0:100 200
enter value for x1 and y1:300 400
```

Experiment 2

AIM:- To implement Bresenham's Line Drawing Algorithm.

ALGORITHM:-

START CO-ORDINATE: (x0, y0)

END CO-ORDINATE: (x0, y0)

STEP 1:

CALCULATE DX & DY

THESE PARAMETERS ARE:

DX = x1 – x0;

DY = y1 – y0;

STEP 2:

CALCULATE DECISION PARAMETER:

P = 2 * DY – DX

STEP 3:

SUPPOSE THE CURRENT POINT IS (xk, yk) AND THE NEXT POINT IS (xk+1, yk+1) THEN FIND THE NEXT POINT USING THE DECISION PARAMETER.

STEP 4:

CONTINUE STEP 3 UNTIL THE ENDPOINT IS REACHED OR THE NO. OF ITERATIONS ARE COMPLETED I.E. THE NUMBER OF ITERATIONS EQUALS (DX – 1)

TWO CASES:

1. IF Pk < 0,

Pnext = Pk +2.DX

Xk+1 = XK+1

Yk+1 = Yk

2. If pk >= 0,

Pnext = Pk + 2DX - 2DY

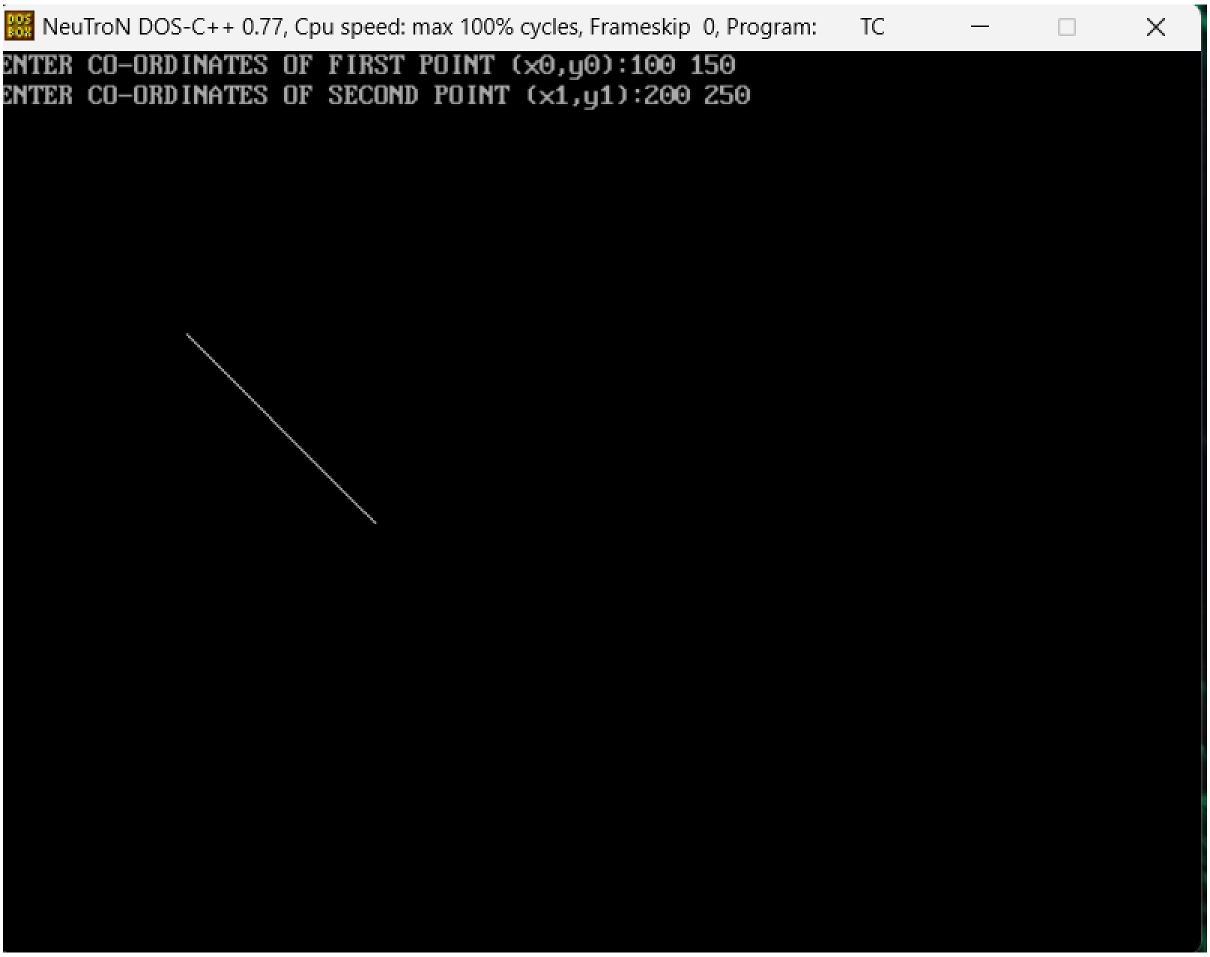
xk+1 = xk+1

yk+1 = yk+1

CODE:-

```
#include <stdio.h>
#include <conio.h>
#include <graphics.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) {
putpixel(x, y, WHITE);
x++;
if (p >= 0) { i
y++;
p = p + 2 * dy - 2 * dx;
} else {
p = p + 2 * dy;
}
}
}
int main() {
int gdriver = DETECT, gmode##;
int x0, y0, x1, y1;
initgraph(&gdriver, &gmode, "C:\\Turboc3\\BGI"); // Adjust path as needed
printf("ENTER CO-ORDINATES OF FIRST POINT (x0 y0): ");
scanf("%d %d", &x0, &y0);
printf("ENTER CO-ORDINATES OF SECOND POINT (x1 y1): ");
scanf("%d %d", &x1, &y1);
drawline(x0, y0, x1, y1);
getch();
closegraph();
return 0;
}
```

OUTPUT:-



NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
ENTER CO-ORDINATES OF FIRST POINT (x0,y0):100 150
ENTER CO-ORDINATES OF SECOND POINT (x1,y1):200 250
```

Experiment 3

AIM:- To implement midpoint circle drawing algorithm in C.

ALGORITHM:-

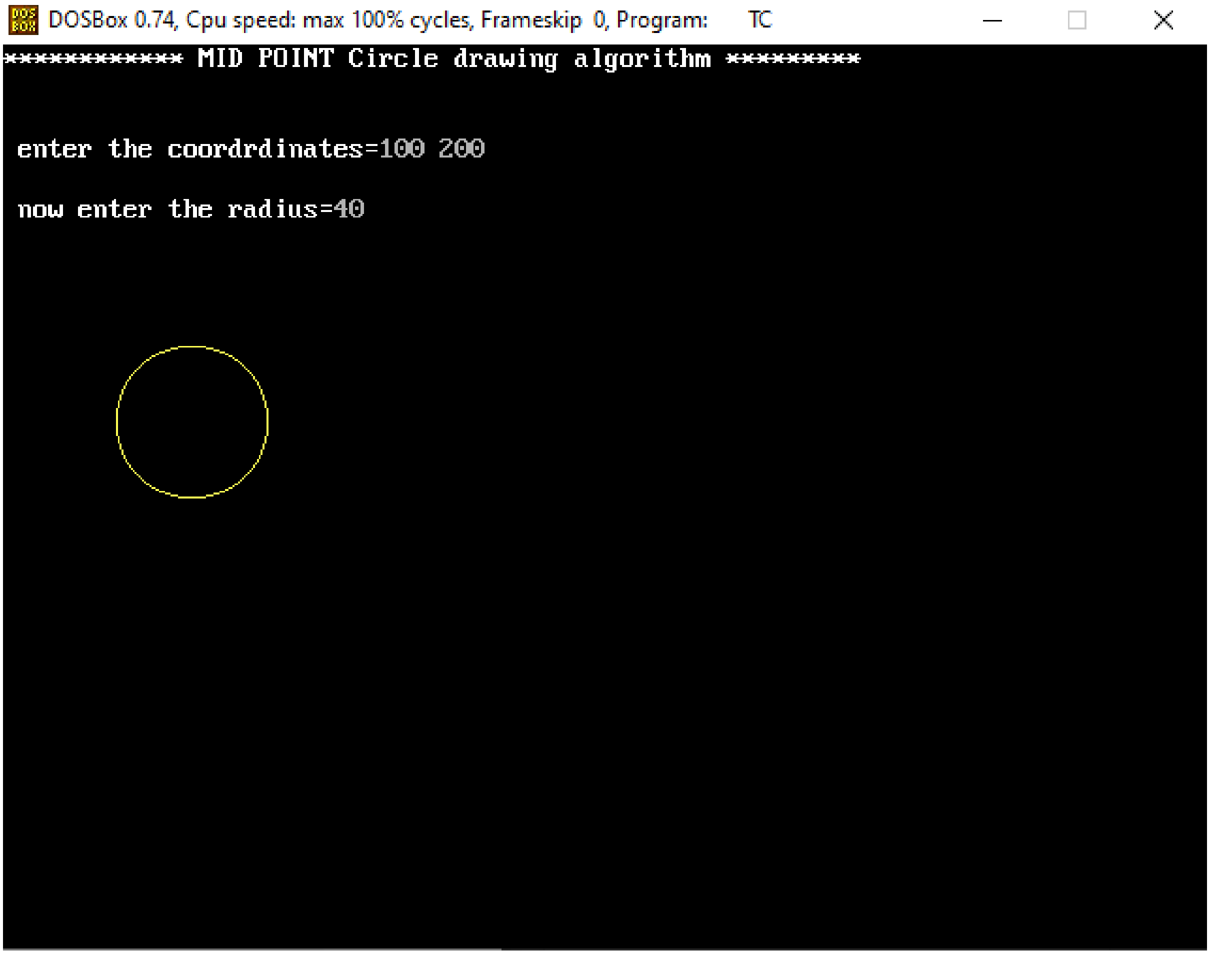
```
Step 1: Put x = 0 and y = r
Step 2: Calculate the initial decision parameter Pk=1-r
Step 3: Plot (x, y)
Step 4: Repeat the steps while x < y
If Pk<0
Pk+1=Pk+2x+3
Xn=X+1
Yn=Y
Else if Pk>0
Pk+1=Pk+2x-2y+5
Xn=X+1
Yn=Y-1
Step 5: Determine symmetry points in the other seven octants.
```

CODE:-

```
#include<graphics.h>

#include<conio.h>

#include<stdio.h>
```

```
void main()

{

int x,y,x_mid,y_mid,radius,dp;

int g_mode,g_driver=DETECT;

clrscr();

initgraph(&g_driver,&g_mode,"C:\\TURBOC3\\BGI");

printf("*********** MID POINT Circle drawing
algorithm

********\n\n");

printf("\n enter the coordinates= ");

scanf("%d %d",&x_mid,&y_mid);

printf("\n now enter the radius =");

scanf("%d",&radius);

x=0;

y=radius;

dp=1-radius;

do

{

putpixel(x_mid+x,y_mid+y,YELLOW);

putpixel(x_mid+y,y_mid+x,YELLOW);

putpixel(x_mid-y,y_mid+x,YELLOW);

putpixel(x_mid-x,y_mid+y,YELLOW);

putpixel(x_mid-x,y_mid-y,YELLOW);

putpixel(x_mid-y,y_mid-x,YELLOW);

putpixel(x_mid+y,y_mid-x,YELLOW);

putpixel(x_mid+x,y_mid-y,YELLOW);

if(dp&lt;0) {

dp+=(2*x)+1;
}
else{
y=y-1;
dp+=(2*x)-(2*y)+1;
}
x=x+1;
}while(y>>x);
getch();
}
```

OUTPUT:-



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
************* MID POINT Circle drawing algorithm **********

 enter the coordrdinates=100 200

 now enter the radius=40
```

Experiment 4

AIM:- To implement midpoint ellipse algorithm in C.

ALGORITHM:-

1.Take input radius along x axis and y axis and obtain center of ellipse.

2.Initially, we assume ellipse to be centered at origin and the first point as : (x, y0)= (0, ry).

3.Obtain the initial decision parameter for region 1 as: p10=ry2+1/4rx2-rx 2ry

4.For every xk position in region 1 :

If p1k<0 then the next point along the is (xk+1 , yk) and p1k+1=p1k+2ry2xk+1+ry2

Else, the next point is (xk+1, yk-1)

And p1k+1=p1k+2ry2xk+1 – 2rx2yk+1+ry2

5.Obtain the initial value in region 2 using the last point (x0, y0) of region 1 as: p20=ry2(x0+1/2)2+rx2 (y0-1)2-rx2ry2

6.At each yk in region 2 starting at k =0 perform the following task.

If p2k>0 the next point is (xk, yk-1) and p2k+1=p2k-2rx2yk+1+rx2

7.Else, the next point is (xk+1, yk -1) and p2k+1=p2k+2ry2xk+1 - 2rx2yk+1+rx2

8.Now obtain the symmetric points in the three quadrants and plot the coordinate value as: x=x+xc, y=y+yc

9.Repeat the steps for region 1 until 2ry2x&gt=2rx2y

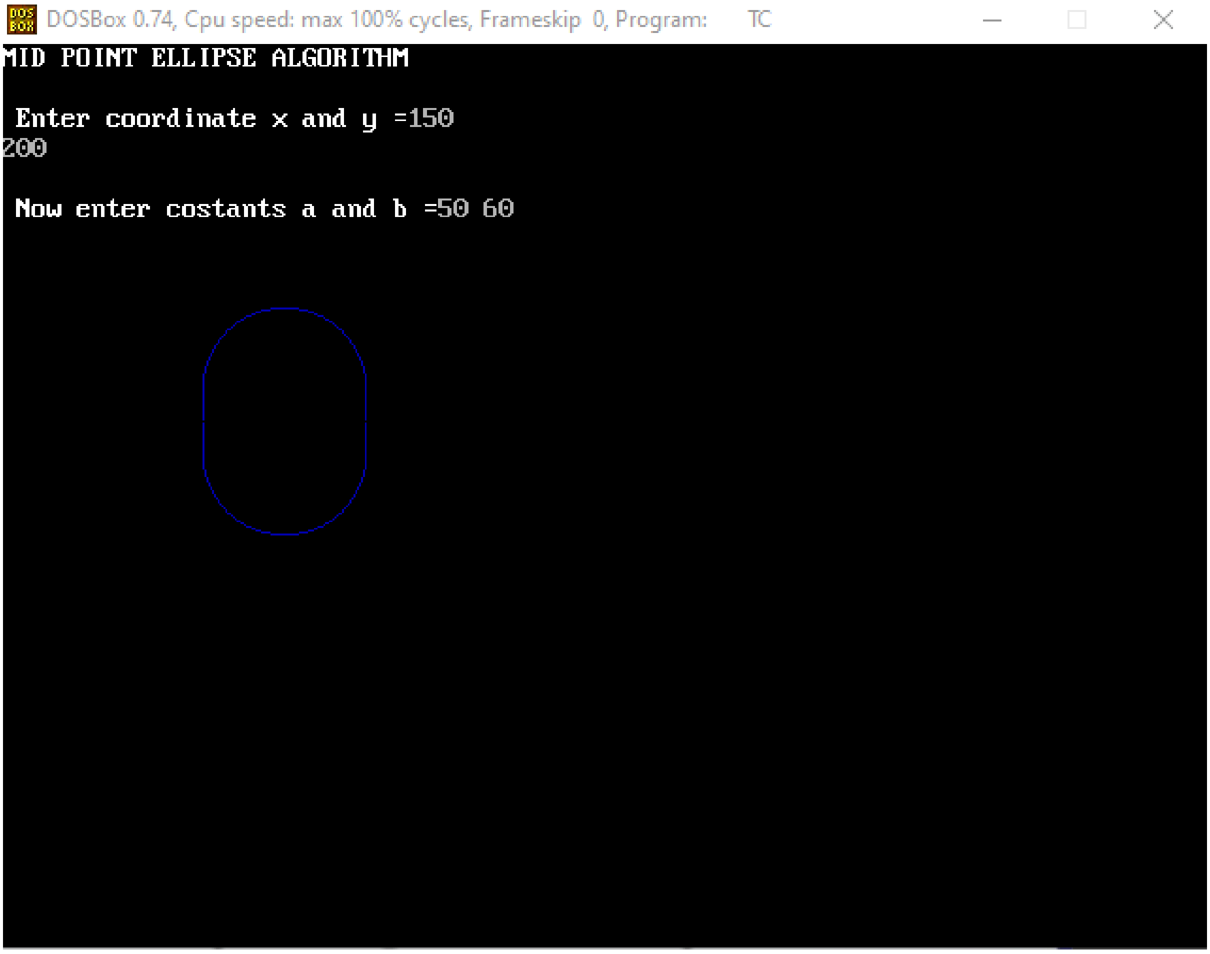
10. Repeat steps for region 2 until y=0

CODE:-

```
#include<stdio.h>
#include<graphics.h>
void main(){
long x,y,x_center,y_center;
long a_sqr,b_sqr, fx,fy, d,a,b,tmp1,tmp2;
int g_driver=DETECT,g_mode;

initgraph(&g_driver,&g_mode,”C:\\TURBOC3\\BGI”);
printf(“ MID POINT ELLIPSE ALGORITHM “);
printf(“\n\n Enter coordinate x and y = “);
scanf(“%ld%ld”,&x_center,&y_center);
printf(“\n Now enter constants a and b = “);
scanf(“%ld%ld”,&a,&b);
x=0;
y=b;
a_sqr=a*a;
b_sqr=b*b;
fx=2*b_sqr*x;
fy=2*a_sqr*y;
d=b_sqr-(a_sqr*b)+(a_sqr*0.25);
do
{
putpixel(x_center+x,y_center+y,1);

putpixel(x_center-x,y_center-y,1);
putpixel(x_center+x,y_center-y,1);
putpixel(x_center-x,y_center+y,1);

if(d<0)
{
d=d+fx+b_sqr;
}
else
{
y=y-1;
d=d+fx+-fy+b_sqr;
fy=fy-(2*a_sqr);
}
x=x+1;
fx=fx+(2*b_sqr);

}
while(fx<fy);
tmp1=(x+0.5)*(x+0.5);
tmp2=(y-1)*(y-1);
d=b_sqr*tmp1+a_sqr*tmp2-(a_sqr*b_sqr);
do
{

putpixel(x_center+x,y_center+y,1);
putpixel(x_center-x,y_center-y,1);
putpixel(x_center+x,y_center-y,1);
putpixel(x_center-x,y_center+y,1);
```

```
if(d>=0)
d=d-fy+a_sqr;
else

{
x=x+1;
d=d+fx-fy+a_sqr;
fx=fx+(2*b_sqr);
}
y=y-1;
fy=fy-(2*a_sqr);
}
while(y>0);
getch();
closegraph();
}
```

OUTPUT:-



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
MID POINT ELLIPSE ALGORITHM

 Enter coordinate x and y =150
200

 Now enter costants a and b =50 60
```

Experiment 5

AIM:- To implement 8 connected flood fill and 8 connected boundary fill algorithm.

ALGORITHM:-

FLOOD FILL:-

```
Procedure floodfill (x, y,fill_ color, old_color: integer)
   If (getpixel (x, y)=old_color)
   {
   setpixel (x, y, fill_color);
   fill (x+1, y, fill_color, old_color);
    fill (x-1, y, fill_color, old_color);
   fill (x, y+1, fill_color, old_color);
   fill (x, y-1, fill_color, old_color);
    }
}
```

BOUNDARY FILL:-

```
void boundaryFill8(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);
      boundaryFill8(x + 1, y, fill_color, boundary_color);
      boundaryFill8(x, y + 1, fill_color, boundary_color);
```

```
        boundaryFill8(x - 1, y, fill_color, boundary_color);
        boundaryFill8(x, y - 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y - 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y + 1, fill_color, boundary_color);
        boundaryFill8(x + 1, y - 1, fill_color, boundary_color);
        boundaryFill8(x + 1, y + 1, fill_color, boundary_color);
    }
}
```

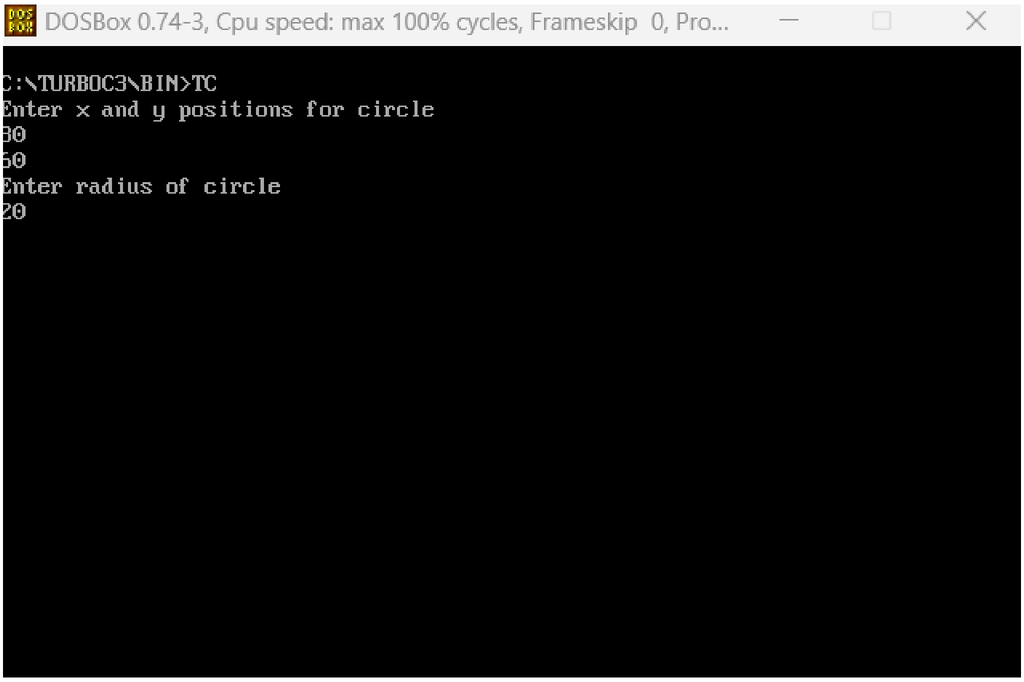
CODE:-

Program for Flood Fill Algorithm in C:-

```
#include<stdio.h>
#include<graphics.h>
#include<dos.h>
void floodFill(int x,int y,int oldcolor,int newcolor)
{
if(getpixel(x,y) == oldcolor)
{
putpixel(x,y,newcolor);
floodFill(x+1,y,oldcolor,newcolor);
floodFill(x,y+1,oldcolor,newcolor);
floodFill(x-1,y,oldcolor,newcolor);
floodFill(x,y-1,oldcolor,newcolor);
}
}
//getpixel(x,y) gives the color of specified pixel
int main()
```

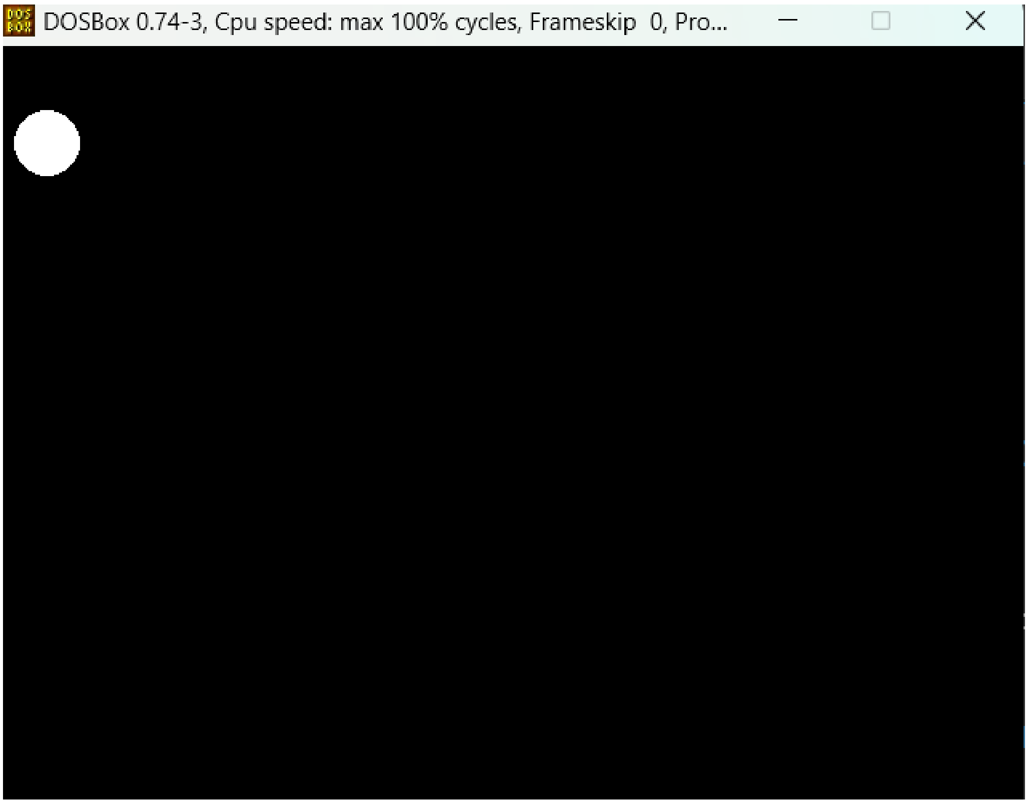
```
{
int gm,gd=DETECT,radius;
int x,y;
printf("Enter x and y positions for circle\n");
scanf("%d%d",&x,&y);
printf("Enter radius of circle\n");
scanf("%d",&radius);
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
circle(x,y,radius);
floodFill(x,y,0,15);
delay(5000);
closegraph();
return 0;
}
```

OUTPUT:-



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

```
C:\TURBOC3\BIN>TC
Enter x and y positions for circle
30
60
Enter radius of circle
20
```



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

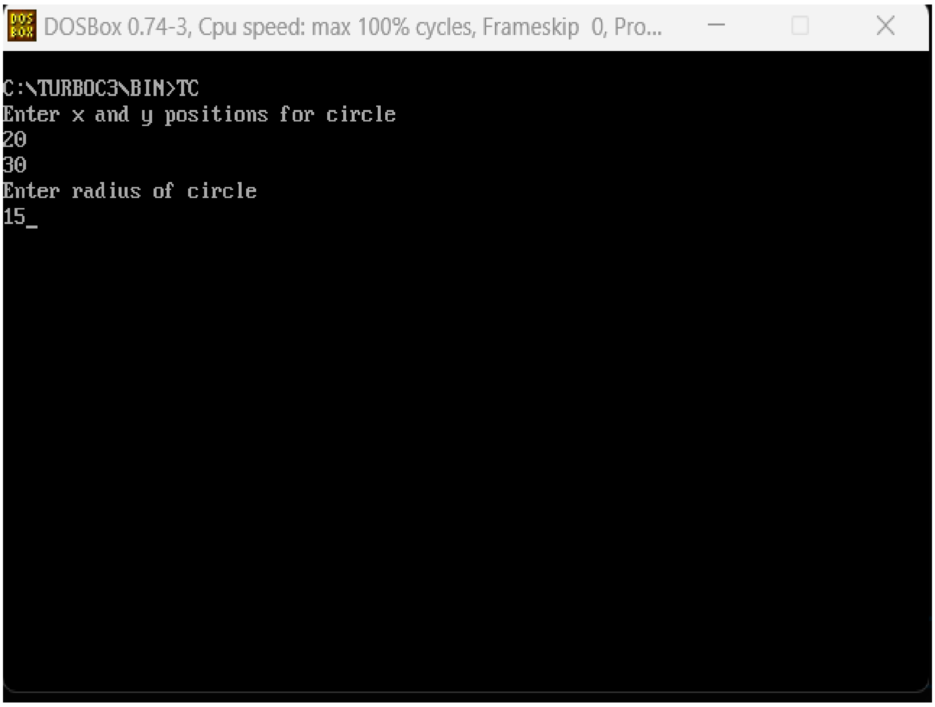
CODE:-

Program for Boundary Fill Algorithm in C:-

```
#include<stdio.h>
```

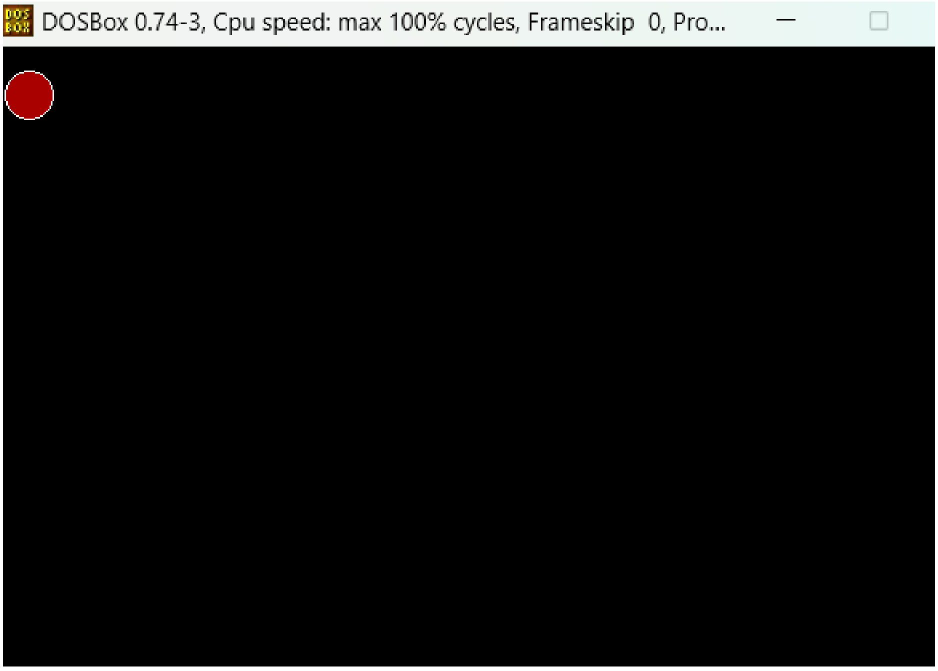
```
#include<graphics.h>
#include<dos.h>
void boundaryfill(int x,int y,int f_color,int
b_color)
{
if(getpixel(x,y)!=b_color && getpixel(x,y)!=f_color)
{
putpixel(x,y,f_color);
boundaryfill(x+1,y,f_color,b_color);
boundaryfill(x,y+1,f_color,b_color);
boundaryfill(x-1,y,f_color,b_color);
boundaryfill(x,y-1,f_color,b_color);
}
}
//getpixel(x,y) gives the color of specified pixel
int main()
{
int gm,gd=DETECT,radius;
int x,y;
printf("Enter x and y positions for circle\n");
scanf("%d%d",&x,&y);
printf("Enter radius of circle\n");
scanf("%d",&radius);
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
circle(x,y,radius);
boundaryfill(x,y,4,15);
delay(5000);
closegraph();
return 0;

}
```

OUTPUT:-



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

```
C:\TURBOC3\BIN>TC
Enter x and y positions for circle
20
30
Enter radius of circle
15_
```



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

Experiment 6

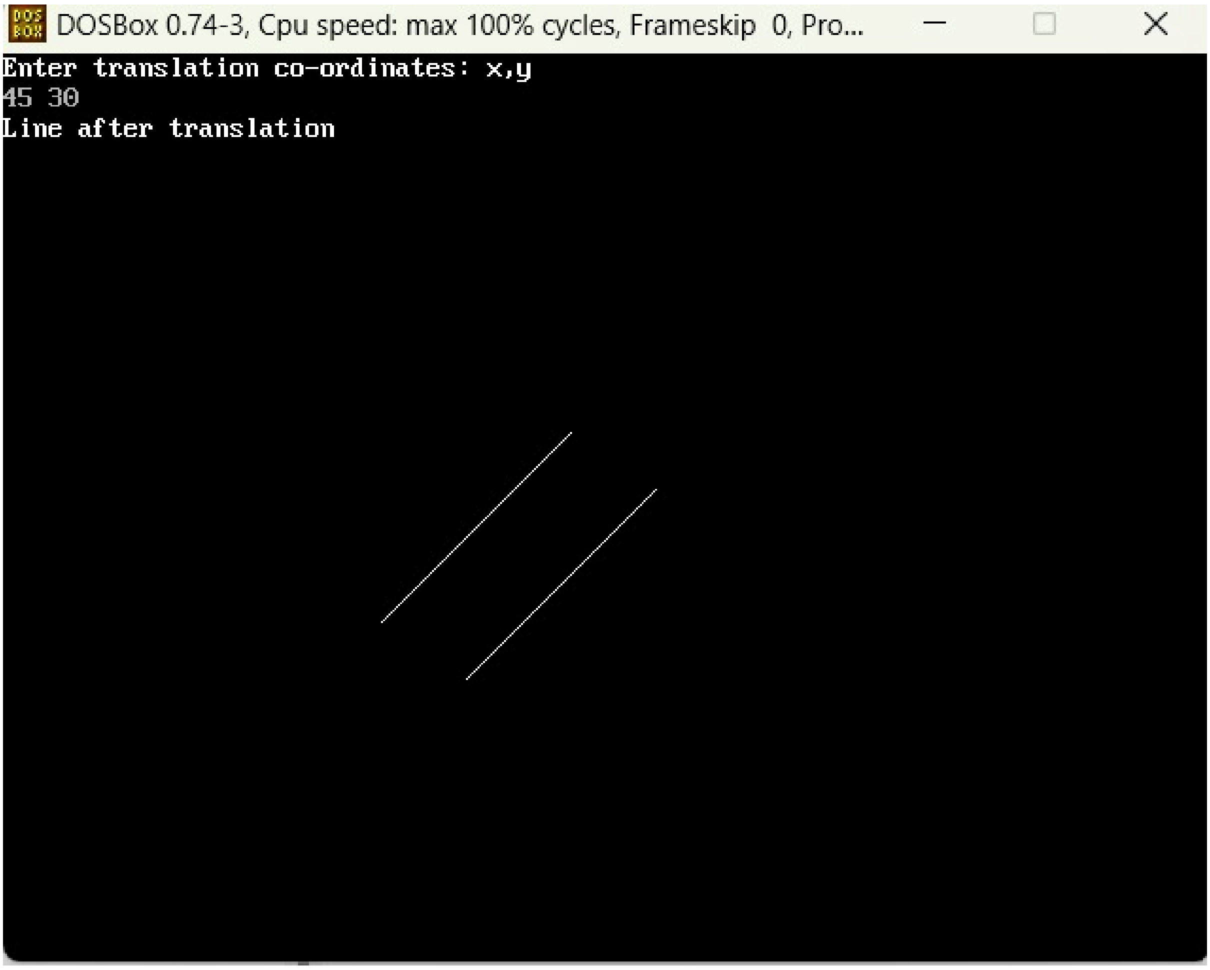
AIM:- To implement 2D transformation operations like Transaltion, Rotation and Scaling in C.

CODE:-

2D Translation:

```
#include&lt;graphics.h&gt;

#include&lt;stdlib.h&gt;

#include&lt;stdio.h&gt;

#include&lt;math.h&gt;

void main()

{

int graphdriver=DETECT,graphmode,errorcode;

int i;

int x2,y2,x1,y1,x,y;

printf(&quot;Enter the 2 line end points:&quot;);

printf(&quot;x1,y1,x2,y2&quot;);

scanf(&quot;%d%d%d%d&quot;,&amp;x1,&amp;y1,&amp;x2,&amp;y2);

initgraph(&amp;graphdriver,&amp;graphmode,&quot;c:\\tc\\bgi&quot;);

line(x1,y1,x2,y2);

printf(&quot;Enter translation co-ordinates &quot;);

printf(&quot;x,y&quot;);

scanf(&quot;%d%d&quot;,&amp;x,&amp;y);
x1=x1+x;
y1=y1+y;
x2=x2+x;
y2=y2+y;
printf(&quot;Line after translation&quot;);
line(x1,y1,x2,y2);
getch();
closegraph();
}
```

OUTPUT:-



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

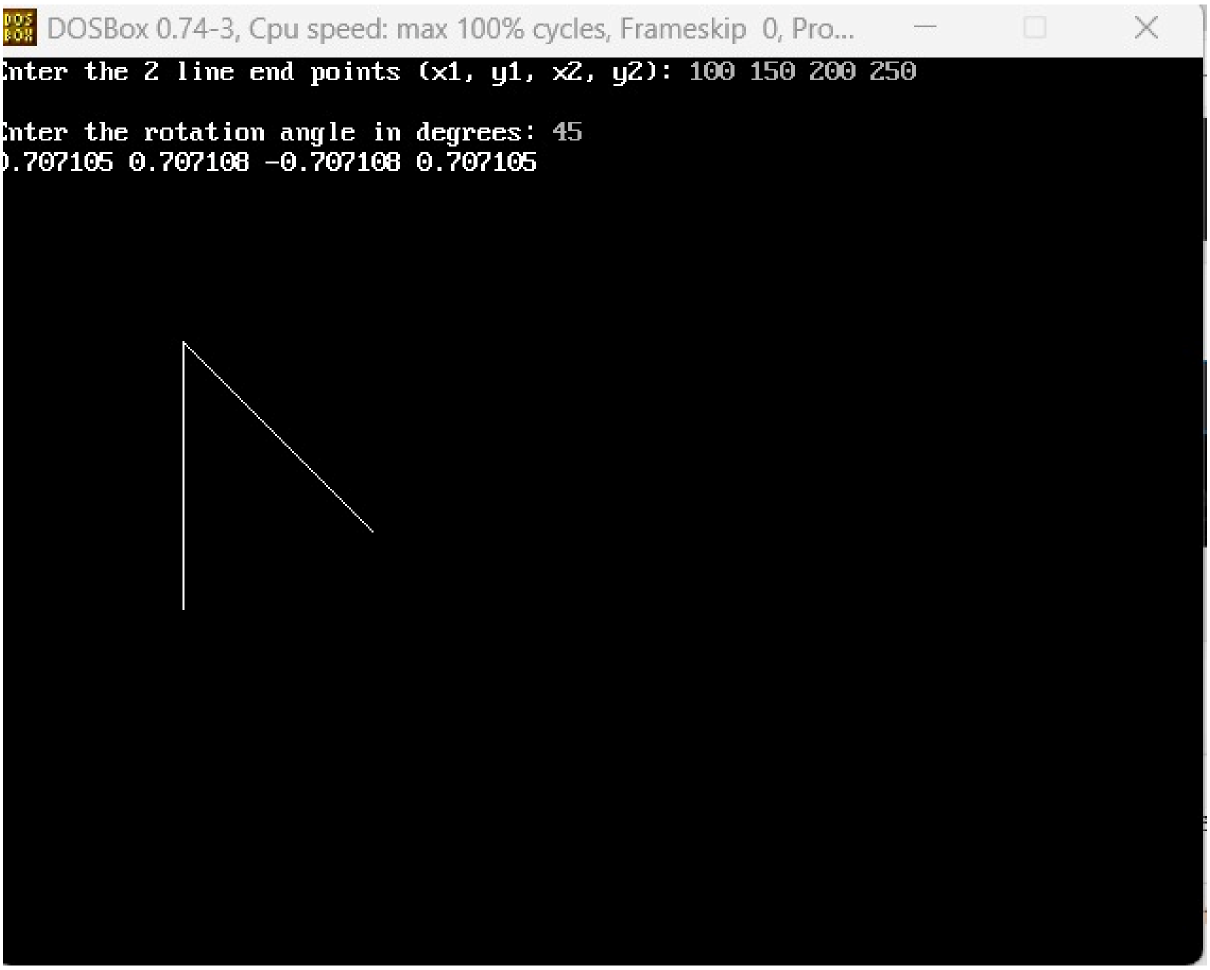
```
Enter translation co-ordinates: x,y
45 30
Line after translation
```

CODE:-

2D Rotation:

```
#include&lt;graphics.h&gt;

#include&lt;stdlib.h&gt;

#include&lt;stdio.h&gt;

#include&lt;math.h&gt;

void main()

{

int graphdriver=DETECT,graphmode,errorcode;

int i;

int x2,y2,x1,y1,x,y,xn,yn;

double r11,r12,r21,r22,th;

clrscr();

printf(&quot;Enter the 2 line end points:&quot;);

printf(&quot;x1,y1,x2,y2&quot;);

scanf(&quot;%d%d%d%d&quot;,&amp;x1,&amp;y1,&amp;x2,&a
mp;y2);

initgraph(&amp;graphdriver,&amp;graphmode,&quot;c:\\t
c\\bgi&quot;);

line(x1,y1,x2,y2);

printf(&quot;\n\n\n[ Enter the angle&quot;);

scanf(&quot;%lf&quot;,&amp;th);

r11=cos((th*3.1428)/180);

r12=sin((th*3.1428)/180);

r21=(-sin((th*3.1428)/180));

r22=cos((th*3.1428)/180);

//printf(&quot;%lf  %lf  %lf  %lf&quot;,r11,r12,r21,r
22);
xn=((x2*r11)-(y2*r12));
yn=((x2*r12)+(y2*r11));
line(x1,y1,xn,yn);
getch();
closegraph();
}
```

OUTPUT:-



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

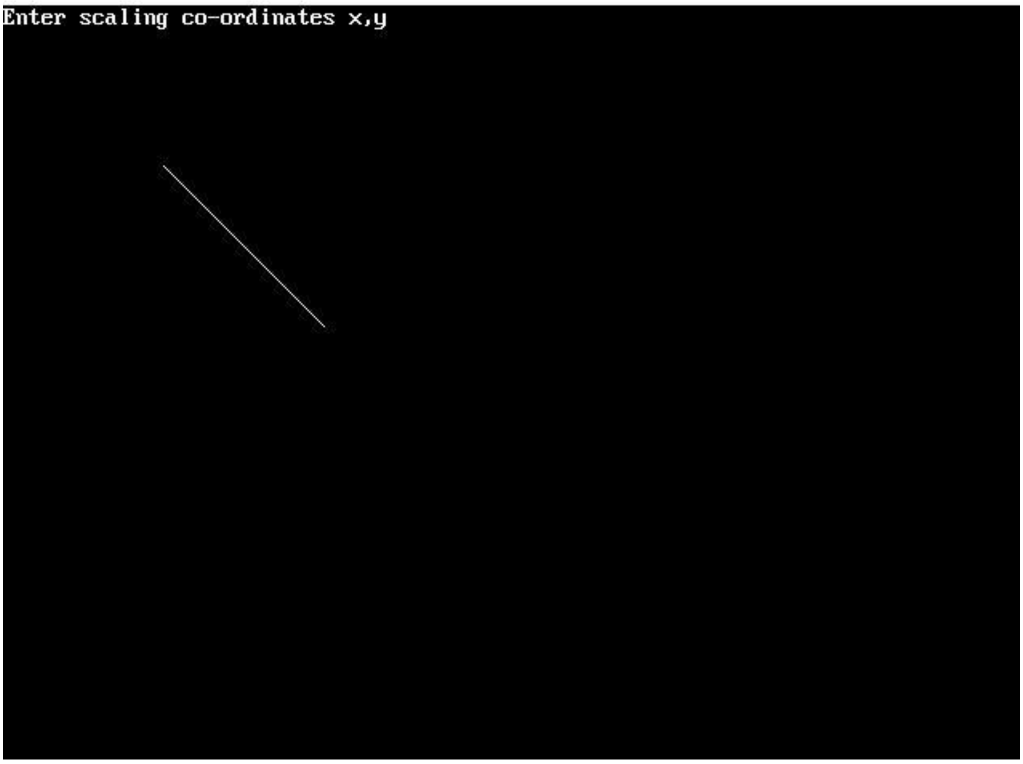
```
:nter the 2 line end points (x1, y1, x2, y2): 100 150 200 250

:nter the rotation angle in degrees: 45
).707105 0.707108 -0.707108 0.707105
```

CODE:-

2D Scaling:

```
#include&lt;graphics.h&gt;

#include&lt;stdlib.h&gt;

#include&lt;stdio.h&gt;

#include&lt;math.h&gt;

void main()

{

int graphdriver=DETECT,graphmode,errorcode;

int i;

int x2,y2,x1,y1,x,y;

printf(&quot;Enter the 2 line end points:&quot;);

printf(&quot;x1,y1,x2,y2&quot;);

scanf(&quot;%d%d%d%d&quot;,&amp;x1,&amp;y1,&amp;x2,&amp;y2);

initgraph(&amp;graphdriver,&amp;graphmode,&quot;c:\\tc\\bgi&quot;);

line(x1,y1,x2,y2);

printf(&quot;Enter scaling co-ordinates &quot;);

printf(&quot;x,y&quot;);

scanf(&quot;%d%d&quot;,&amp;x,&amp;y);

x1=(x1*x);

y1=(y1*y);

x2=(x2*x);

y2=(y2*y);

printf(&quot;Line after scaling&quot;);

line(x1,y1,x2,y2);

getch();
closegraph();
}
```

OUTPUT:-



```
Enter scaling co-ordinates x,y
```

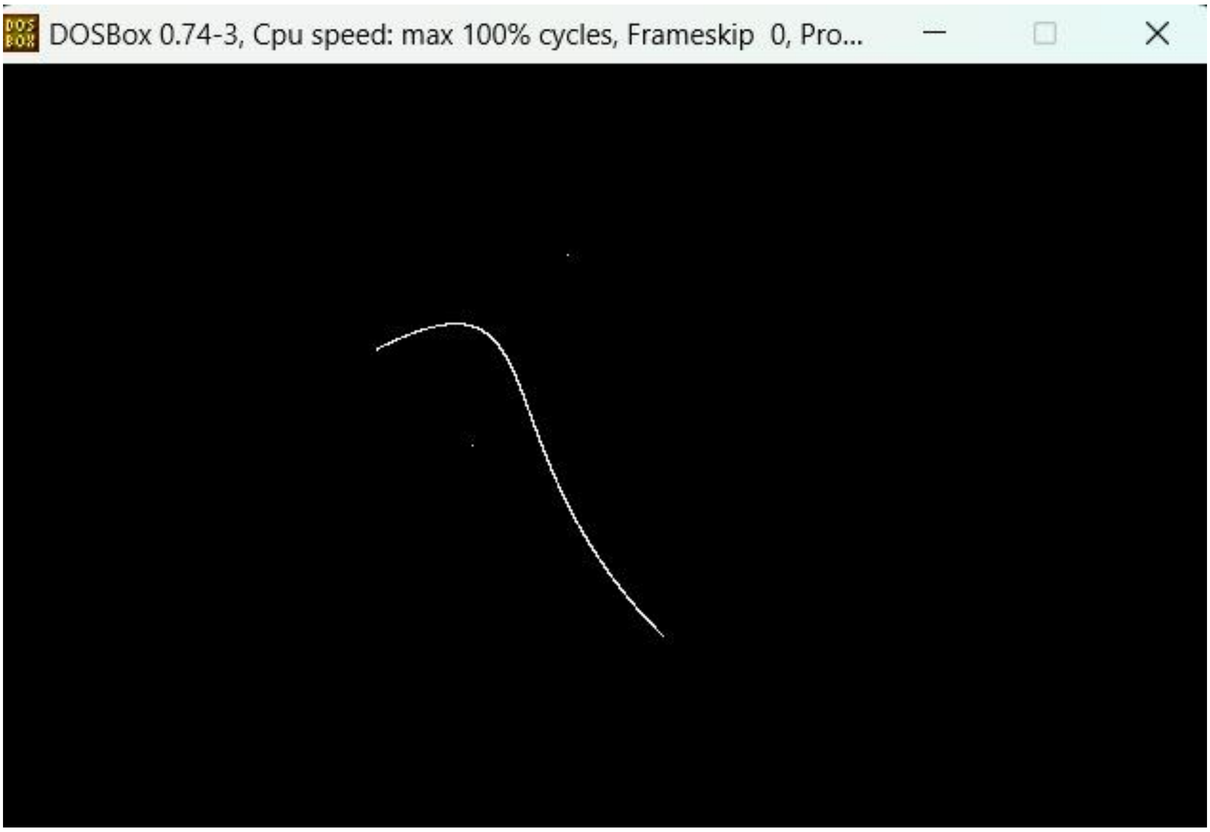
Experiment 7

AIM:- To implement Cubic Bezier Curve in C.

CODE:-

```
#include <stdio.h>

#include <graphics.h>

#include <math.h>

#include<conio.h>


int x[4]={200,300,250,350};

int y[4]={150,100,200,300};


void bezier ()

{

int i;

double t,xt,yt;

for (t = 0.0; t < 1.0; t += 0.0005)

{

xt = pow(1-t,3)x[0]+3*t*pow(1-t,2)*x[1]+3*pow(t,2)(1-
t)*x[2]+pow(t,3)*x[3];

yt = pow(1-t,3)y[0]+3*t*pow(1-t,2)*y[1]+3*pow(t,2)(1-
t)*y[2]+pow(t,3)*y[3];

putpixel (xt, yt,WHITE);

}
for (i=0; i<4; i++)
 putpixel (x[i], y[i], YELLOW);
getch();
closegraph();
}

void main()
{
int gd = DETECT, gm;
initgraph (&gd, &gm, "..\\bgi");
bezier ();
}
```

OUTPUT:-



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

EXPREIMENT 8

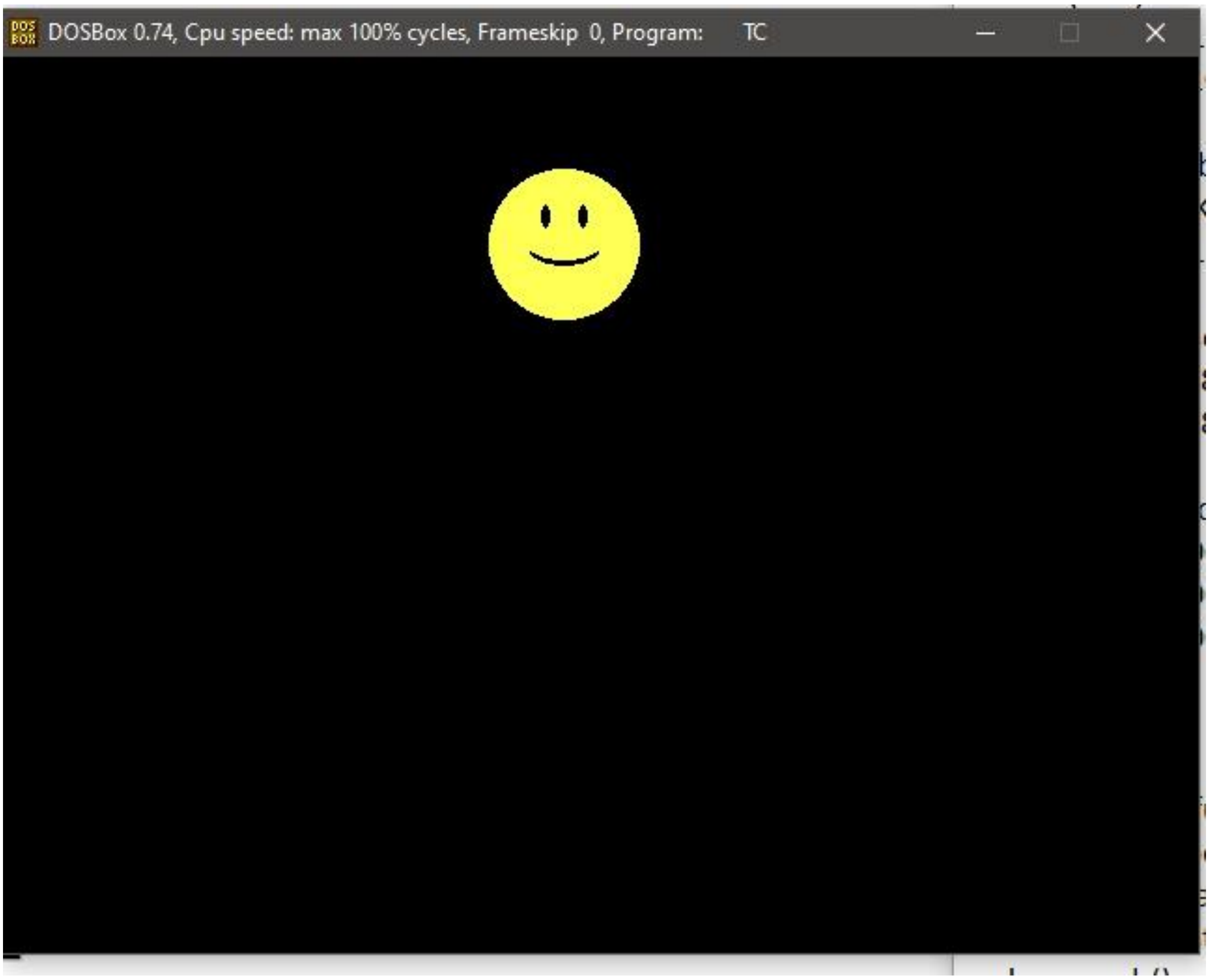
AIM:- Write a program in C to perform Animation (such as Rising Sun, Moving Vehicle, Smileys, Screen saver etc.)

CODE:-

```
// C program to create a smiley face
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <stdio.h>

// Driver Code
int main()
{

    // Initialize graphic driver
    int gr = DETECT, gm;

    // Initialize graphics mode by passing
    // three arguments to initgraph function

    // &gdriver is the address of gdriver
    // variable, &gmode is the address of
    // gmode and "C:\\Turboc3\\BGI" is the
// directory path where BGI files
// are stored
initgraph(&gr, &gm, "C:\\Turboc3\\BGI");

// Set color of smiley to yellow
setcolor(YELLOW);

// creating circle and fill it with
// yellow color using floodfill.
circle(300, 100, 40);
setfillstyle(SOLID_FILL, YELLOW);
floodfill(300, 100, YELLOW);

// Set color of background to black
setcolor(BLACK);
setfillstyle(SOLID_FILL, BLACK);

// Use fill ellipse for creating eyes
fillellipse(310, 85, 2, 6);
fillellipse(290, 85, 2, 6);

// Use ellipse for creating mouth
ellipse(300, 100, 205, 335, 20, 9);
ellipse(300, 100, 205, 335, 20, 10);
ellipse(300, 100, 205, 335, 20, 11);

getch();
```

```
    // closegraph function closes the
    // graphics mode and deallocates
    // all memory allocated by
    // graphics system
    closegraph();

    return 0;
}
```

OUTPUT:-


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

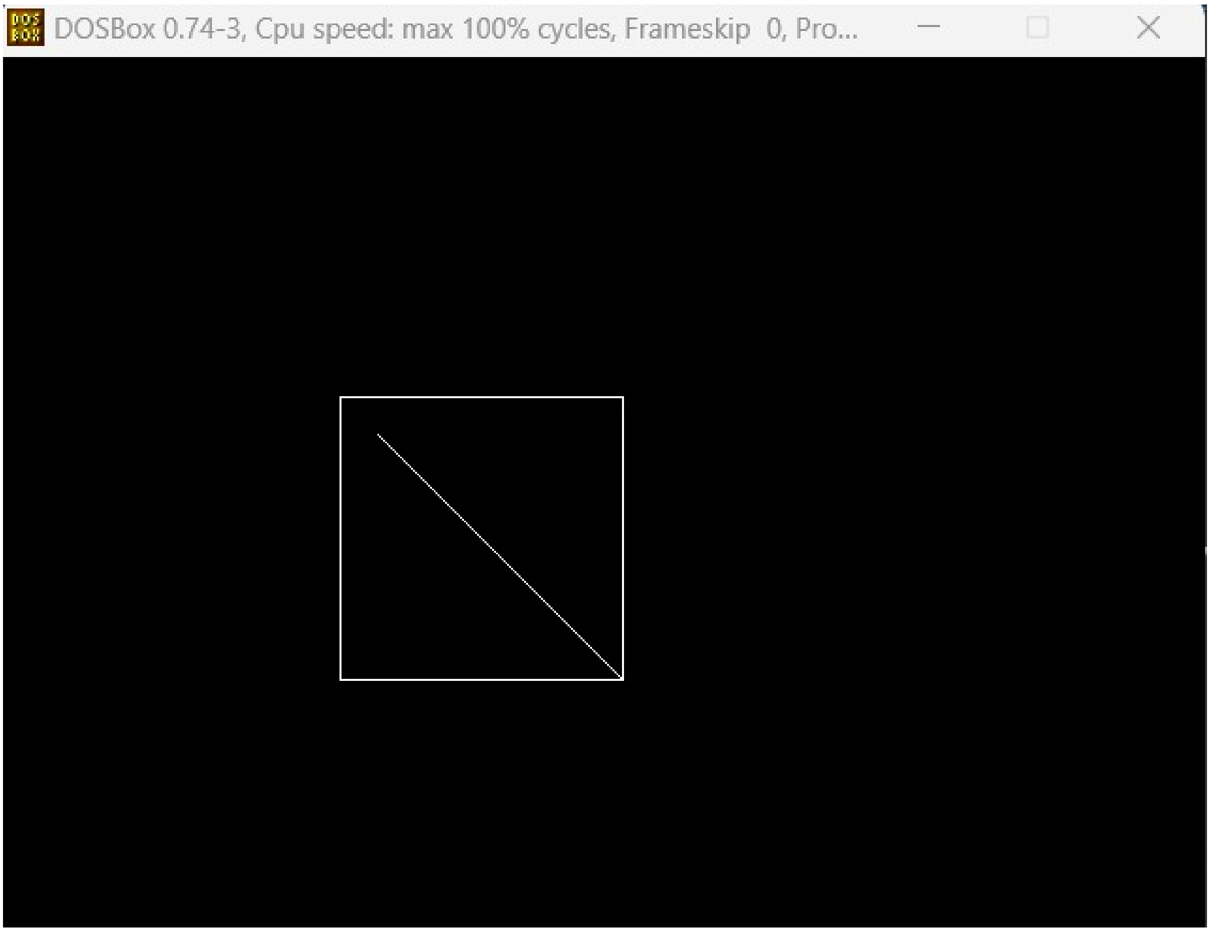
Experiment 9

AIM:- Write a program to implement Liang Barsky line clipping algorithm in C.

CODE:-

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

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

    initgraph(&gd, &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:-


DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

Experiment 10

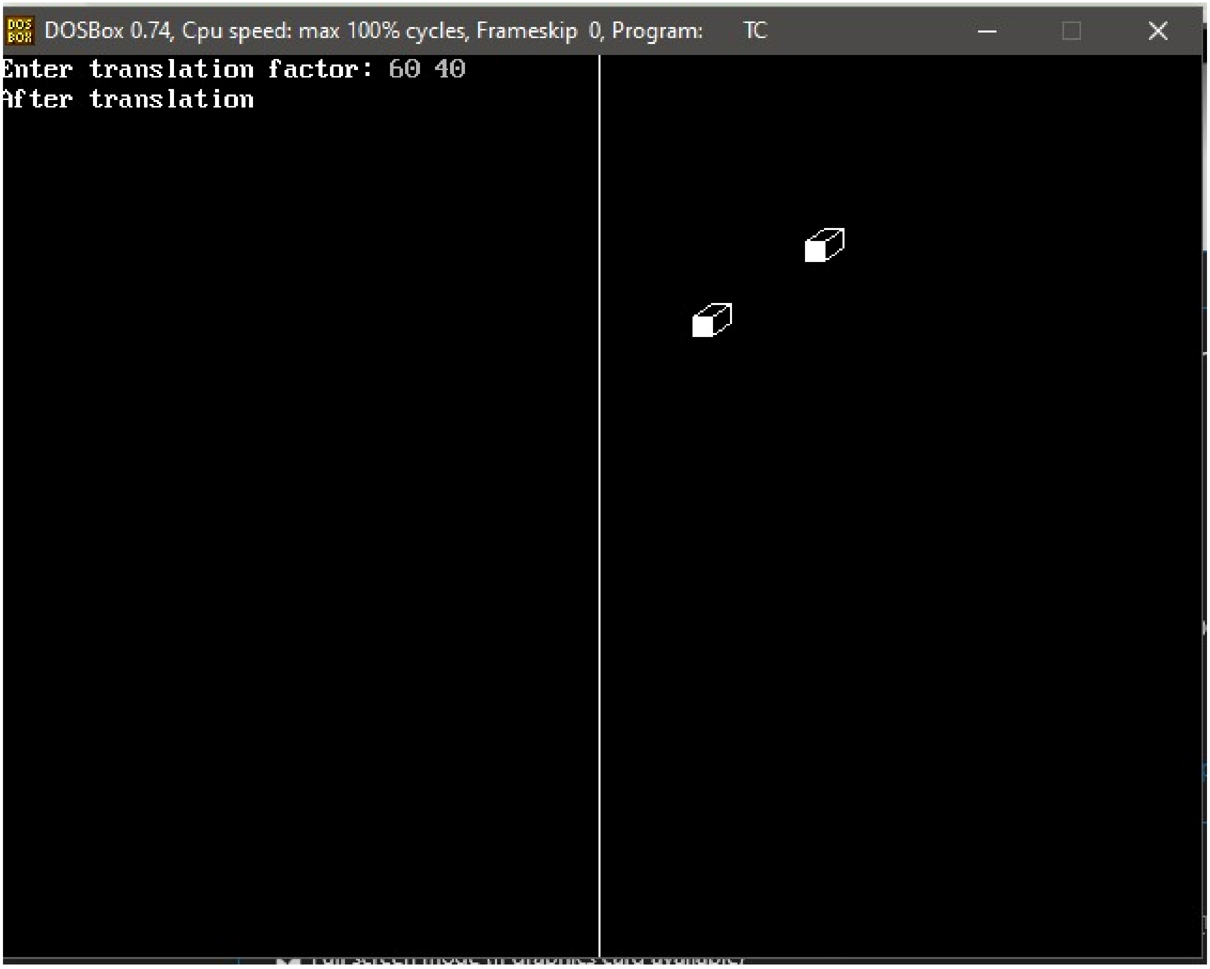
AIM:- To implement 3D Transformation operations such as Translation and Scaling in C.

CODE:-

3D Translation:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int maxx,maxy,midx,midy;
void axis()
{
getch();
cleardevice();
line(midx,0,midx,maxy);
line(0,midy,maxx,midy);
}
void main()
{
int x,y,z,o,x1,x2,y1,y2;
int gd=DETECT,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"c:\\tc\\bgi");
//setfillstyle(0,getmaxcolor());
maxx=getmaxx();
maxy=getmaxy();
midx=maxx/2;
midy=maxy/2;

axis();

bar3d(midx+50,midy-100,midx+60,midy-90,10,1);

printf("Enter translation factor");
scanf("%d%d",&x,&y);
//axis();
printf("After translation:");
bar3d(midx+x+50,midy-(y+100),midx+x+60,midy-
(y+90),10,1);
getch();
closegraph();
}
```

OUTPUT:-



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

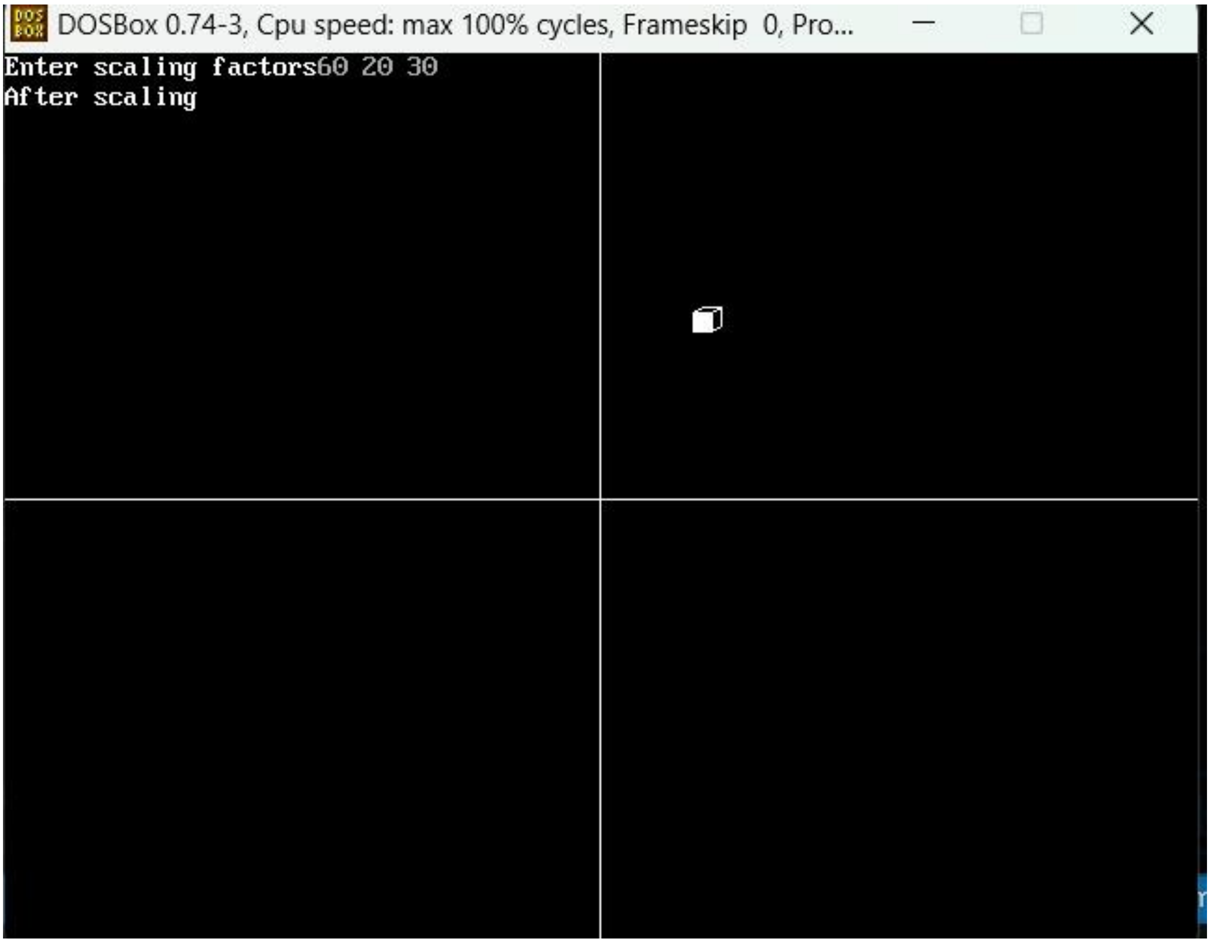
```
Enter translation factor: 60 40
After translation
```

CODE:-

3D Scaling:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int maxx,maxy,midx,midy;
void axis()
{
getch();
cleardevice();
line(midx,0,midx,maxy);
line(0,midy,maxx,midy);
}
void main()
{
int x,y,z,o,x1,x2,y1,y2;
int gd=DETECT,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"c:\\tc\\bgi");
//setfillstyle(0,getmaxcolor());
maxx=getmaxx();
maxy=getmaxy();
midx=maxx/2;
midy=maxy/2;
axis();
bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
printf("Enter scaling factors");
scanf("%d%d%d", &x,&y,&z);
//axis();
printf("After scaling");
bar3d(midx+(x*50),midy-(y*100),midx+(x*60),midy-
(y*90),5*z,1);
//axis();
getch();
closegraph();
}
```

OUTPUT:-



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

```
Enter scaling factors60 20 30
After scaling
```