INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN

Computer Science & Engineering Department

Computer Graphics and multimedia Lab

LAB FILE

Submitted to:

Miss Vibha Pratap

Submitted by:

Sakshi

09101012018

B.Tech CSE2

EXPERIMENT-1

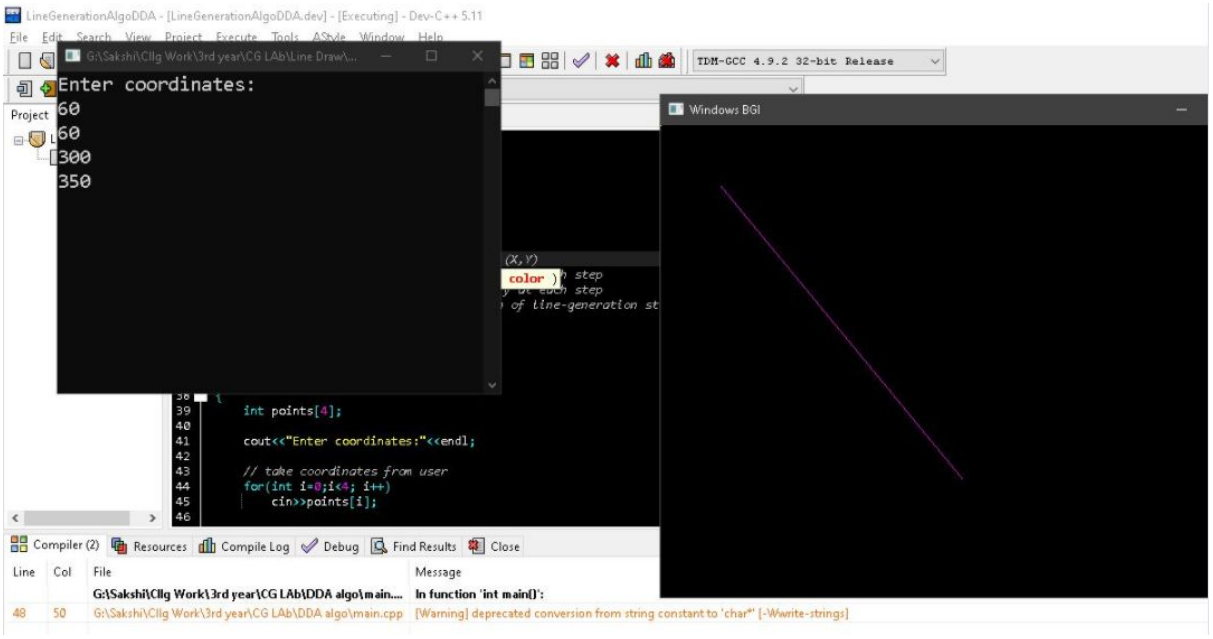
Aim: Write a program to implement the DDA algorithm.

Code:

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

int abs (int n)
{
	return ( (n>0) ? n : ( n * (-1)));
}

void DDA(int X0, int Y0, int X1, int Y1)
{
	// how much to be moved
	int dx = X1 - X0;
	int dy = Y1 - Y0;

	// calculate steps required for generating pixels
	int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);

	// calculate increment in x & y for each steps
	float Xinc = dx / (float) steps;
	float Yinc = dy / (float) steps;

	// Put pixel for each step
	float X = X0;
	float Y = Y0;
	for (int i = 0; i <= steps; i++)
	{
		putpixel(X,Y,5); // put pixel at (X,Y)
		X += Xinc;		// increment in x at each step
		Y += Yinc;		// increment in y at each step
		delay(10);	// for visualization of line-generation step by step
	}
	return;
}

int main()
{
	int points[4];

	cout<<"Enter coordinates:"<<endl;
```

```
        // take coordinates from user
        for(int i=0;i<4; i++)
                cin>>points[i];

        int gd = DETECT, gm;
        initgraph (&gd, &gm, "C:\\Dev-Cpp\\MinGW64\\lib");

        DDA(points[0], points[1], points[2], points[3]);

        getch();
        return 0;
}
```

Output:



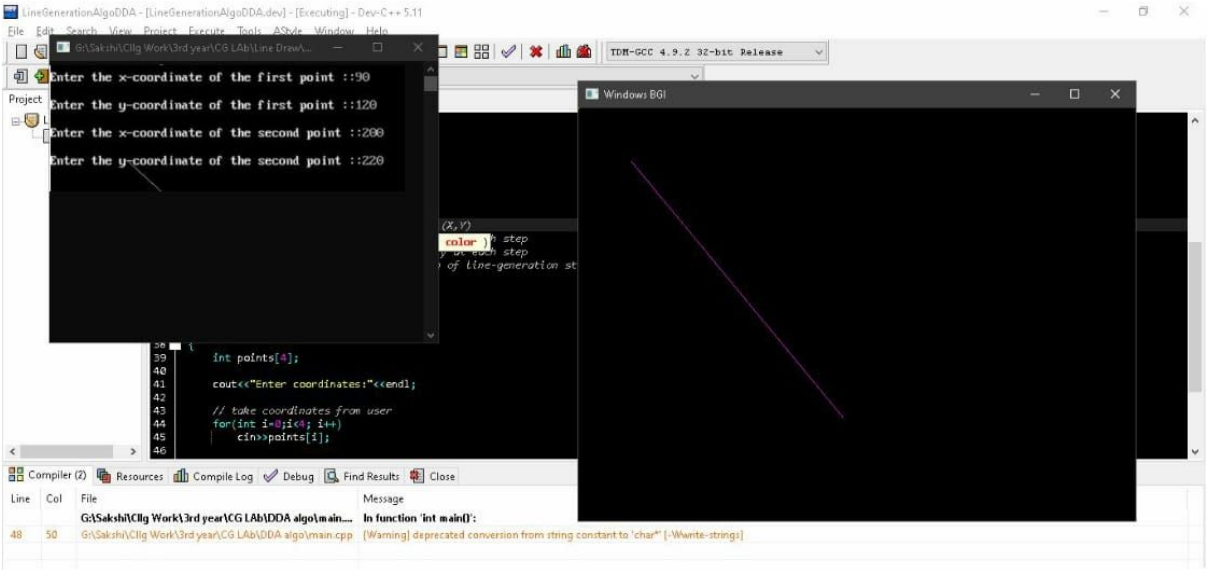
EXPERIMENT-2

Aim: Write a program to implement Bresenham's line algorithm.

Code:

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

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++)
{
        cout << "(" << x << "," << y << ")\n";

        slope_error_new += m_new;

        if (slope_error_new >= 0)
        {
                y++;
                slope_error_new -= 2 * (x2 - x1);
        }
}
}

int main()
{
int x1 = 3, y1 = 2, x2 = 15, y2 = 5;
bresenham(x1, y1, x2, y2);
return 0;
}
```

Output:


LineGenerationAlgoDDA - [LineGenerationAlgoDDA.dev] - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
G:\Sakshi\CIlg Work\3rd year\CG LAb\Line Draw\...
Enter the x-coordinate of the first point ::90
Enter the y-coordinate of the second point ::120
Enter the x-coordinate of the second point ::200
Enter the y-coordinate of the second point ::220
Windows BGI
TDM-GCC 4.9.2 32-bit Release
Project
int points[4];
cout<<"Enter coordinates:"<<endl;
// take coordinates from user
for(int i=0;i<4; i++)
cin>>points[i];
Compiler (2) Resources Compile Log Debug Find Results Close
Line Col File Message
G:\Sakshi\CIlg Work\3rd year\CG LAb\DDA algo\main.... In function 'int main()':
48 50 G:\Sakshi\CIlg Work\3rd year\CG LAb\DDA algo\main.cpp [Warning] deprecated conversion from string constant to 'char*' [-Wwrite-strings]

EXPERIMENT-3

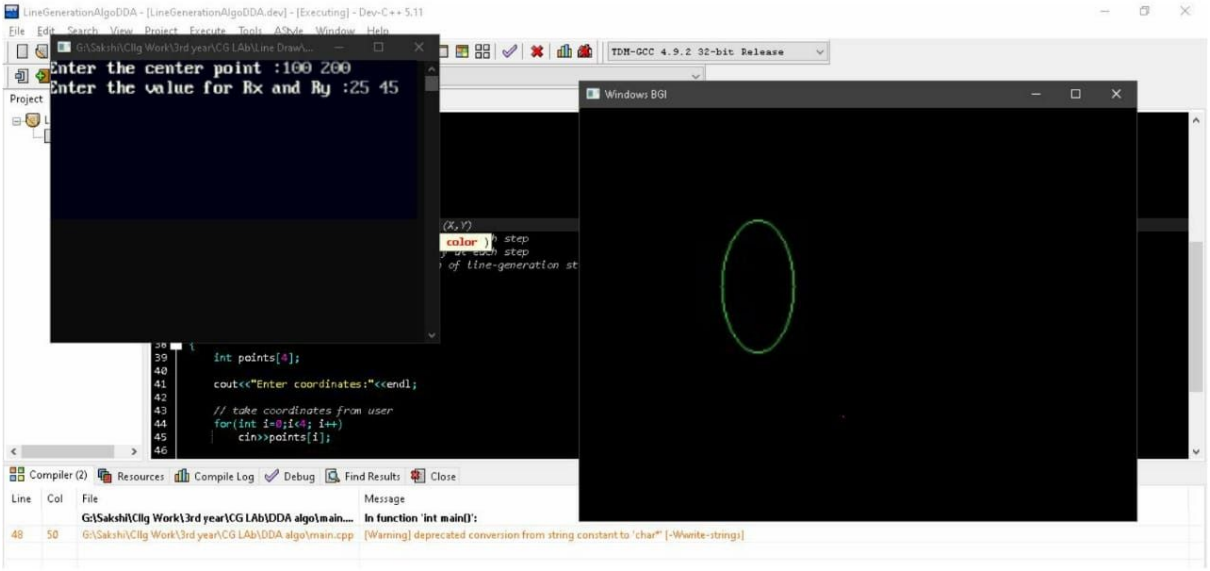
Aim: Write a program to implement Midpoint ellipse generating algorithm

Code:

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

void midptellipse(int rx, int ry,
                  int xc, int yc)
{
    float dx, dy, d1, d2, x, y;
    x = 0;
    y = ry;

    d1 = (ry * ry) - (rx * rx * ry) + (0.25 * rx * rx);
    dx = 2 * ry * ry * x;
    dy = 2 * rx * rx * y;

    while (dx < dy)
    {

        cout << x + xc << " , " << y + yc << endl;
        cout << -x + xc << " , " << y + yc << endl;
        cout << x + xc << " , " << -y + yc << endl;
        cout << -x + xc << " , " << -y + yc << endl;

        if (d1 < 0)
        {
            x++;
            dx = dx + (2 * ry * ry);
            d1 = d1 + dx + (ry * ry);
        }
        else
        {
            x++;
            y--;
            dx = dx + (2 * ry * ry);
            dy = dy - (2 * rx * rx);
            d1 = d1 + dx - dy + (ry * ry);
        }
    }

    d2 = ((ry * ry) * ((x + 0.5) * (x + 0.5))) +
        ((rx * rx) * ((y - 1) * (y - 1))) -
        (rx * rx * ry * ry);

    while (y >= 0)
```

```
        {

                cout << x + xc << " , " << y + yc << endl;
                cout << -x + xc << " , " << y + yc << endl;
                cout << x + xc << " , " << -y + yc << endl;
                cout << -x + xc << " , " << -y + yc << endl;

                if (d2 > 0)
                {
                        y--;
                        dy = dy - (2 * rx * rx);
                        d2 = d2 + (rx * rx) - dy;
                }
                else
                {
                        y--;
                        x++;
                        dx = dx + (2 * ry * ry);
                        dy = dy - (2 * rx * rx);
                        d2 = d2 + dx - dy + (rx * rx);
                }
        }
}

int main()
{
        midptellipse(25, 45, 100, 200);

        return 0;
}
```

Output:


LineGenerationAlgoDDA - [LineGenerationAlgoDDA.dev] - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 32-bit Release
Enter the center point :100 200
Enter the value for Rx and Ry :25 45
Windows BGI
int points[8];
cout<<"Enter coordinates:"<<endl;
// take coordinates from user
for(int i=0;i<4; i++)
cin>>points[i];
Compiler (2) Resources Compile Log Debug Find Results Close
Line Col File Message
G:\Sakshi\Cllg Work\3rd year\CG LAb\DDA algo\main.... In function 'int main()':
48 50 G:\Sakshi\Cllg Work\3rd year\CG LAb\DDA algo\main.cpp [Warning] deprecated conversion from string constant to 'char*' [-Wwrite-strings]

EXPERIMENT-4

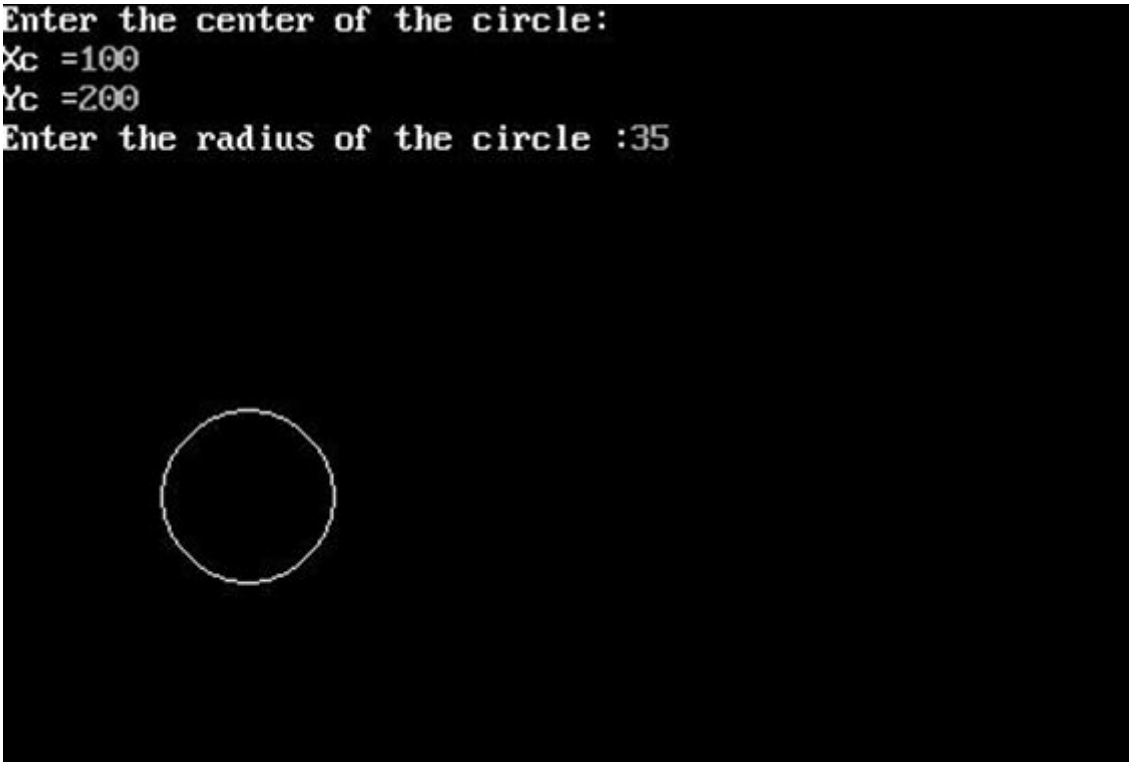
Aim: Write a program to implement Midpoint circle generating algorithm

Code:

```
#include<graphics.h>
#include<stdio.h>
void pixel(int xc,int yc,int x,int y);
int main()
{
	int gd,gm,xc,yc,r,x,y,p;
	detectgraph(&gd,&gm);
	initgraph(&gd,&gm,"C://TurboC3//BGI");

	scanf("%d%d",&xc,&yc);
	printf("Enter radius of circle :");
	scanf("%d",&r);

	x=0;
	y=r;
	p=1-r;
	pixel(xc,yc,x,y);

	while(x<y)
	{
		if(p<0)
		{
			x++;
			p=p+2*x+1;
		}
		else
		{
			x++;
			y--;
			p=p+2*(x-y)+1;
		}
		pixel(xc,yc,x,y);
	}

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

void pixel(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);
}
```

Output:



EXPERIMENT-5

Aim: Write a program to implement Bresenham's circle generating algorithm.

Code:

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

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

void circleBres(int xc, int yc, int r)
{
	int x = 0, y = r;
	int d = 3 - 2 * r;
	drawCircle(xc, yc, x, y);
	while (y >= x)
	{
		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 = 300, yc = 300, r2 = 50;
	int gd = DETECT, gm;
	initgraph(&gd, &gm, "");
```

```
        circleBres(xc, yc, r);
        return 0;
}
```

Output:

EXPERIMENT-6

Aim: Write a program to implement Line Clipping Algorithm using Cohen Sutherland Algorithm.

Code:

```
#include<iostream.h>
#include<stdlib.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);
```

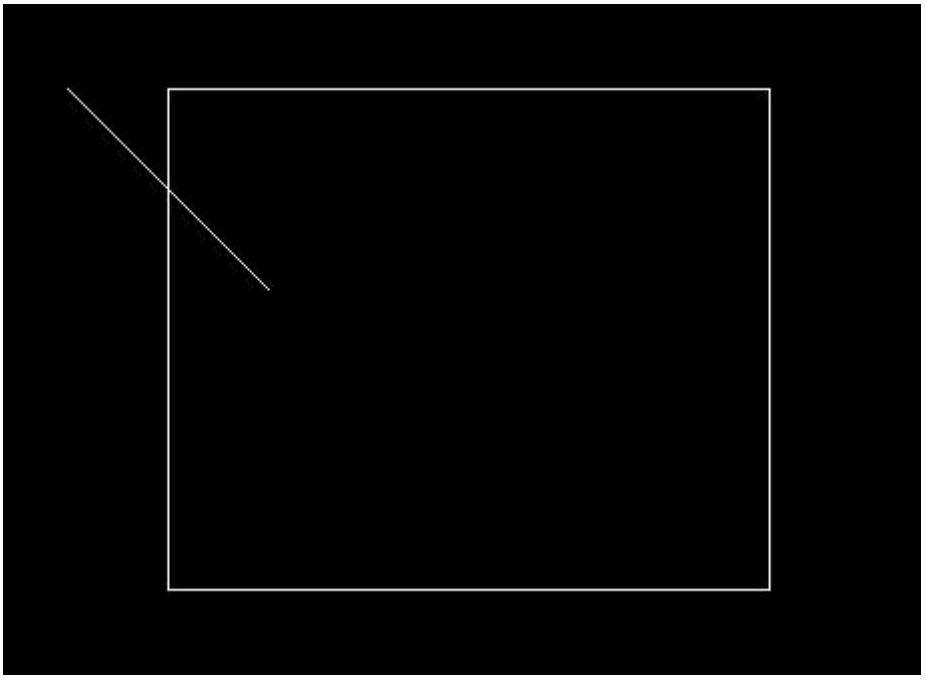
```
                        drawline(p1,p2);
                        break;
        case 1:drawwindow();
                        delay(500);
                        break;
        case 2:p3=resetendpt(p1,p2);
                        p4=resetendpt(p2,p1);
                        drawwindow();
       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

        delay(5000);
        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];
```

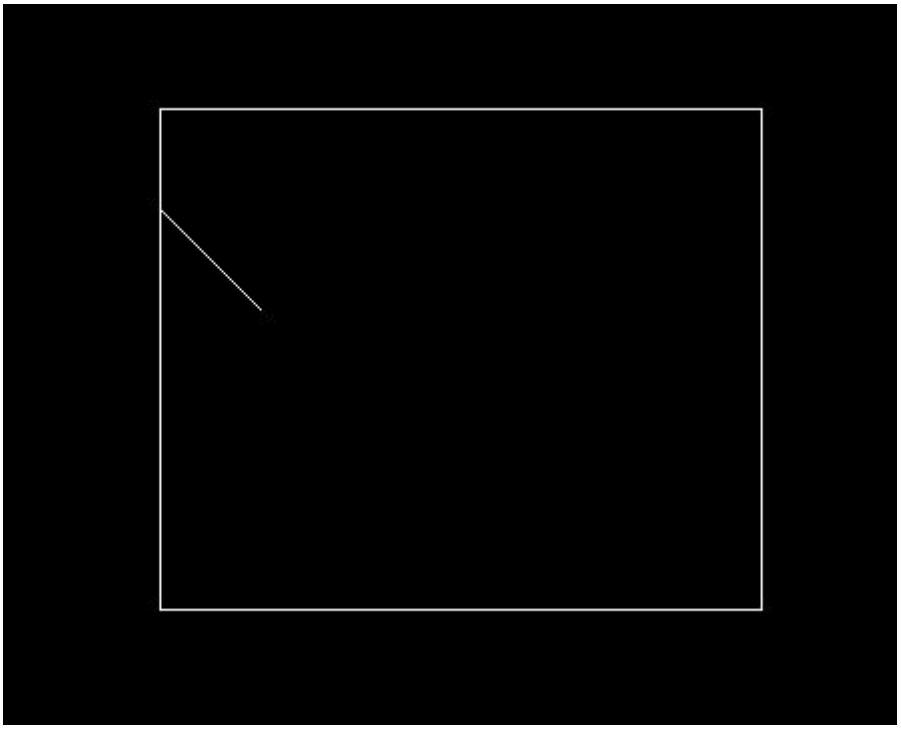
```
                if(temp.y<=350 && temp.y>=100)
                        return (temp);
        }

        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:

Before Clipping:



After Clipping:



EXPERIMENT-7

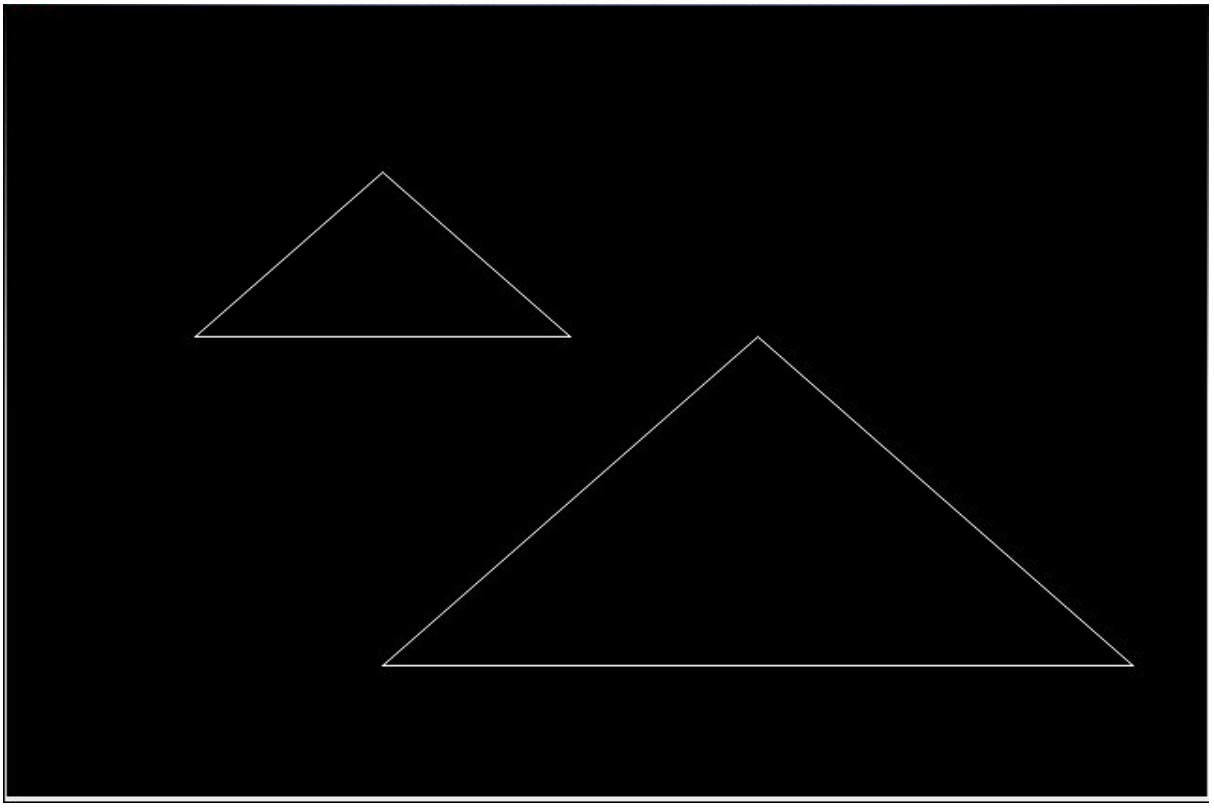
Aim: Write a program to scale a polygon.

Code:

```
#include<stdio.h>
#include<graphics.h>
void findNewCoordinate(int s[][2], int p[][1])
{
	int temp[2][1] = { 0 };

	for (int i = 0; i < 2; i++)
		for (int j = 0; j < 1; j++)
			for (int k = 0; k < 2; k++)
				temp[i][j] += (s[i][k] * p[k][j]);

	p[0][0] = temp[0][0];
	p[1][0] = temp[1][0];
}

void scale(int x[], int y[], int sx, int sy)
{

	line(x[0], y[0], x[1], y[1]);
	line(x[1], y[1], x[2], y[2]);
	line(x[2], y[2], x[0], y[0]);

	int s[2][2] = { sx, 0, 0, sy };
	int p[2][1];

	for (int i = 0; i < 3; i++)
	{
		p[0][0] = x[i];
		p[1][0] = y[i];

		findNewCoordinate(s, p);

		x[i] = p[0][0];
		y[i] = p[1][0];
	}

	line(x[0], y[0], x[1], y[1]);
	line(x[1], y[1], x[2], y[2]);
	line(x[2], y[2], x[0], y[0]);
}

int main()
{
```

```
        int x[] = { 100, 200, 300 };
        int y[] = { 200, 100, 200 };
        int sx = 2, sy = 2;

        int gd, gm;
        detectgraph(&gd, &gm);
        initgraph(&gd, &gm," ");

        scale(x, y, sx,sy);
        getch();

        return 0;
}
```

Output:



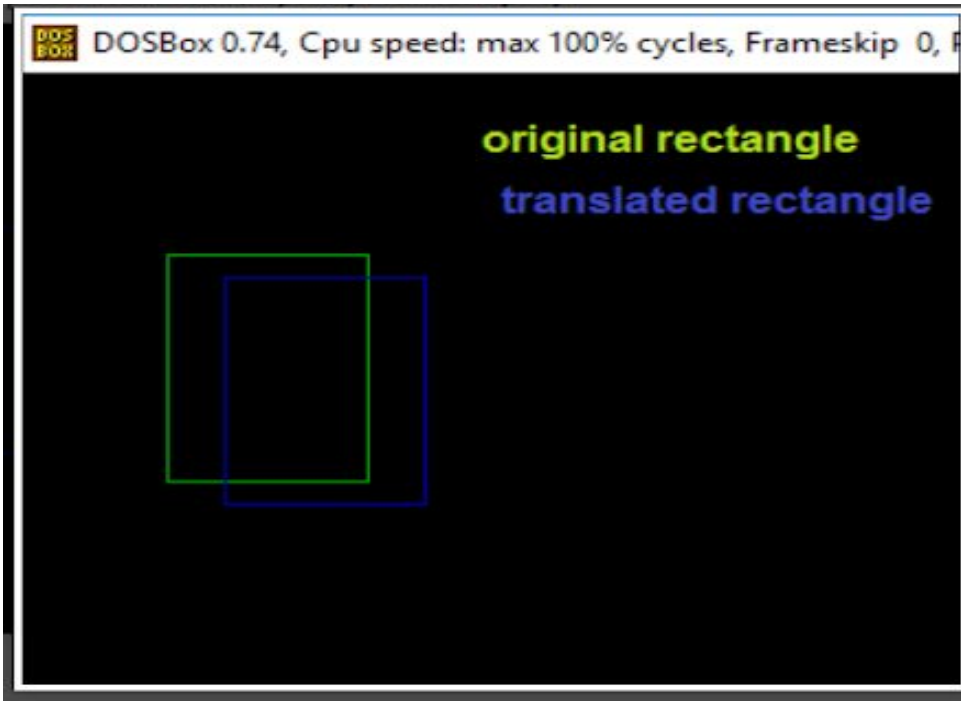
EXPERIMENT-8

Aim: Write a program to translate a polygon.

Code:

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

void translateRectangle ( int P[][2], int T[])
{
        int gd = DETECT, gm, errorcode;
        initgraph (&gd, &gm, "c:\\tc\\bgi");
        setcolor (2);
        rectangle (P[0][0], P[0][1], P[1][0], P[1][1]);
        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];
        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};
        translateRectangle (P, T);
        return 0;
}
```

Output:


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0,
original rectangle
translated rectangle

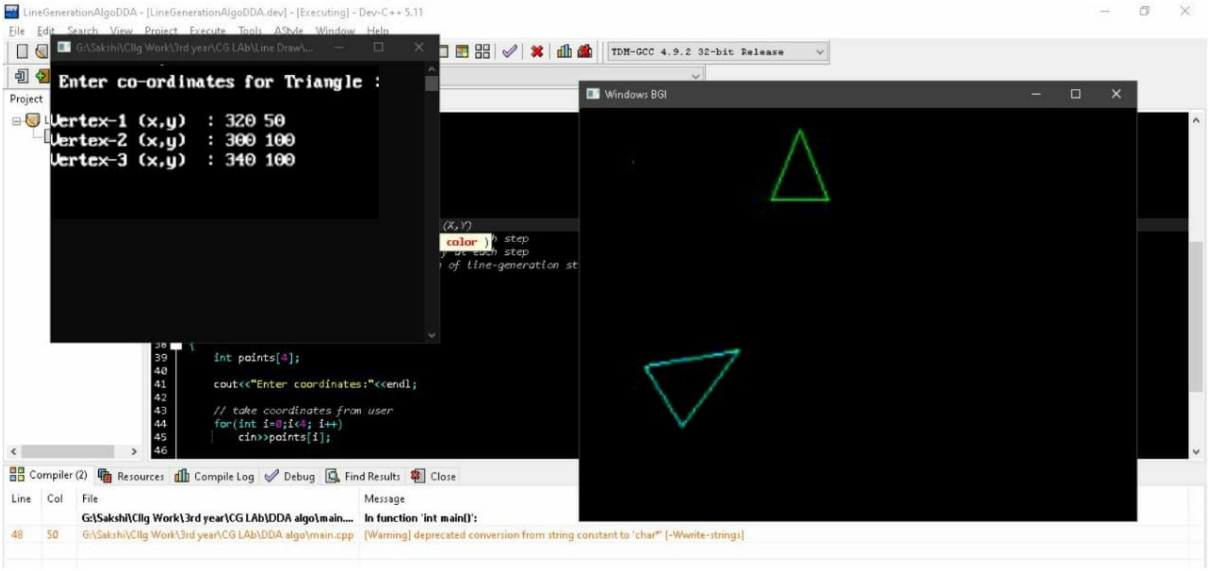
EXPERIMENT-9

Aim: Write a program to rotate a polygon.

Code:

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

int main(void)
{
 int poly[8], rpoly[8],angle;
 int gd = DETECT, gm;
 initgraph(&gd,&gm,"C:\\TC\\BGI");

 printf("\n Enter co-ordinates for Triangle :  \n\n");

 printf("Vertex 1 (x,y)  : ");
 scanf("%d %d", &poly[0], &poly[1]);
 printf("Vertex 2 (x,y)  : ");
 scanf("%d %d", &poly[2], &poly[3]);
 printf("Vertex 3 (x,y)  : ");
 scanf("%d %d", &poly[4], &poly[5]);
 poly[6] = poly[0];
 poly[7] = poly[1];

 setcolor(2);
 drawpoly(4,poly);

 angle = 45;
 rpoly[0] = poly[0]*cos(angle) - poly[1]*sin(angle);
```

Output:


LineGenerationAlgoDDA - [LineGenerationAlgoDDA.dev] - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 32-bit Release
Project
G:\Sakshi\Cllg Work\3rd year\CG LAb\Line Draw\...
Enter co-ordinates for Triangle :
Vertex-1 (x,y) : 320 50
Vertex-2 (x,y) : 300 100
Vertex-3 (x,y) : 340 100
Windows BGI
int points[8];
cout<<"Enter coordinates:"<<endl;
// take coordinates from user
for(int i=0;i<4; i++)
cin>>points[i];
Compiler (2) Resources Compile Log Debug Find Results Close
Line Col File Message
G:\Sakshi\Cllg Work\3rd year\CG LAb\DDA algo\main.... In function 'int main()':
48 50 G:\Sakshi\Cllg Work\3rd year\CG LAb\DDA algo\main.cpp [Warning] deprecated conversion from string constant to 'char*' [-Wwrite-strings]

EXPERIMENT-10

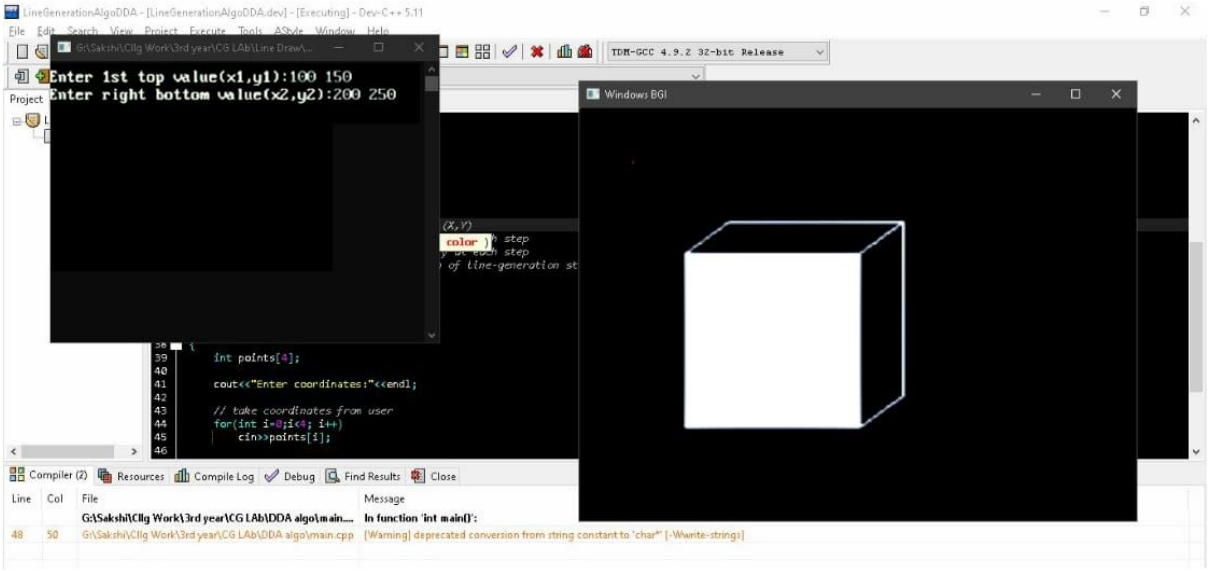
Aim: Write a program to implement basic transformations (scaling, rotation, translation) on a 3D object

Code:

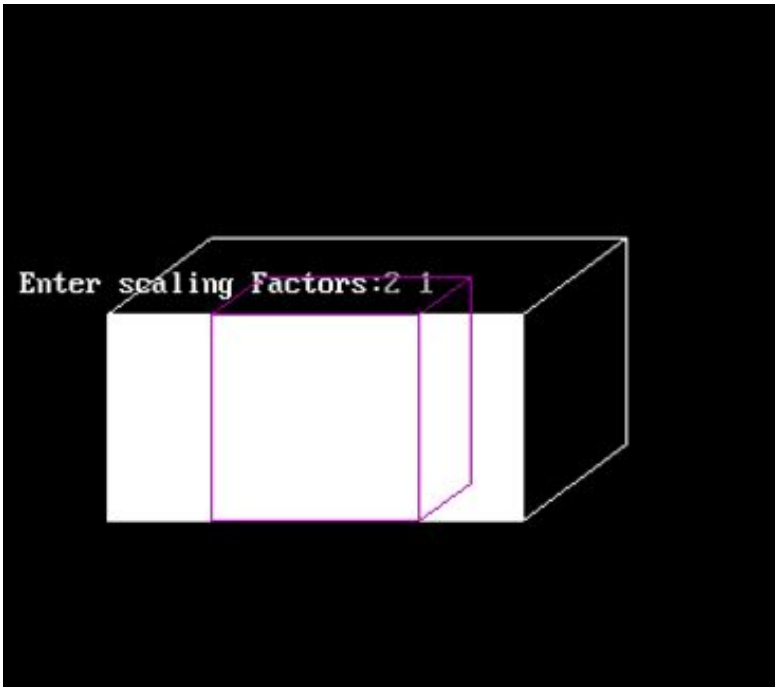
a) Scaling of a 3D Object:

```
#include stdio.h
#include graphics.h
#include math.h
void scale();

int maxx,maxy,midx,midy;
void main()
{
        int ch;
        int gd=DETECT,gm;
        detectgraph(&gd,&gm);
        initgraph(&gd,&gm," ");
        scale();
return 0;
}

void scale()
{
        int x,y,z,o,x1,x2,y1,y2;
        midx=200;
        midy=200;
        bar3d(midx+50,midy-100,midx+100,midy-50,20,0);
        printf("before scaling\n");
        printf("Enter scaling factors\n");
        scanf("%d %d %d", &x,&y,&z);
        printf("After scaling\n");
        bar3d(midx+(x*50),midy-(y*100),midx+(x*100),midy-(y*50),20*z,1);
}
```

Output:


Enter 1st top value(x1,y1):100 150
Enter right bottom value(x2,y2):200 250
Windows BGI
int points[4];
cout<<"Enter coordinates:"<<endl;
// take coordinates from user
for(int i=0;i<4; i++)
cin>>points[i];
Compiler (2)
Resources
Compile Log
Debug
Find Results
Close
Line
Col
File
Message
In function 'int main()':
[Warning] deprecated conversion from string constant to 'char*' [-Wwrite-strings]


Enter the center point :100 200
Enter the value for Rx and Ry :25 45



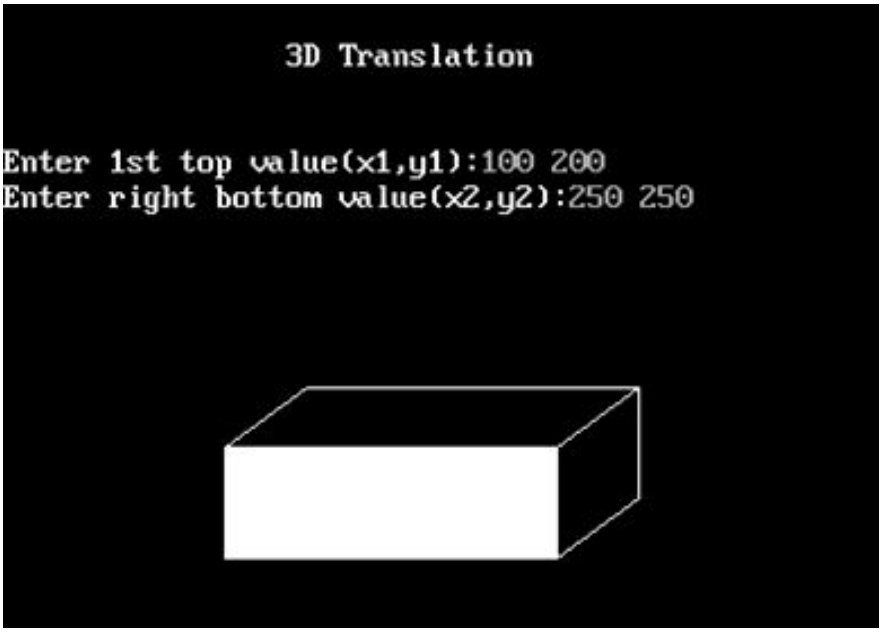
b) Translation of a 3D object:

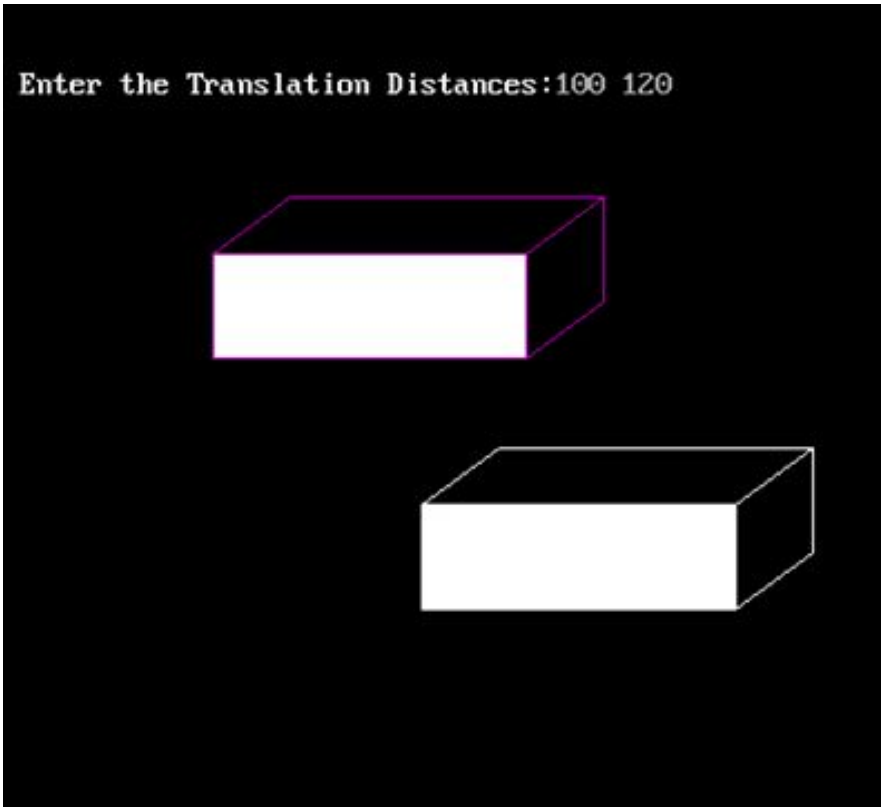
Code:

```
#include stdio.h
#include graphics.h
#include math.h

void trans();

int maxx,maxy,midx,midy;

void main()

{

        int ch;

        int gd=DETECT,gm;

        detectgraph(&gd,&gm);

        initgraph(&gd,&gm," ");

        trans();
return 0;
}

void trans()

{
```

```
        int x,y,z,o,x1,x2,y1,y2;

        midx=200;

        midy=200;

        bar3d(midx+50,midy-100,midx+100,midy-50,20,1);
        delay(1000);
        printf("Enter translation factor");

        scanf("%d%d",&x,&y);

        printf("After translation:");

        bar3d(midx+x+50,midy-(y+100),midx+x+100,midy-(y+50),20,1);

}
```

Output:





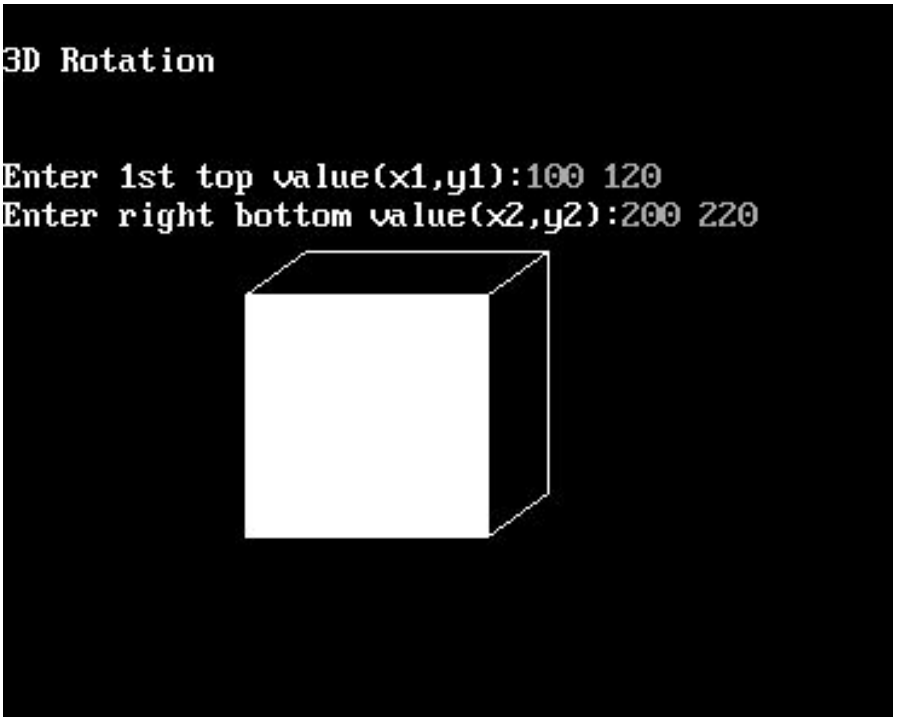
c) Rotation of a 3D object:

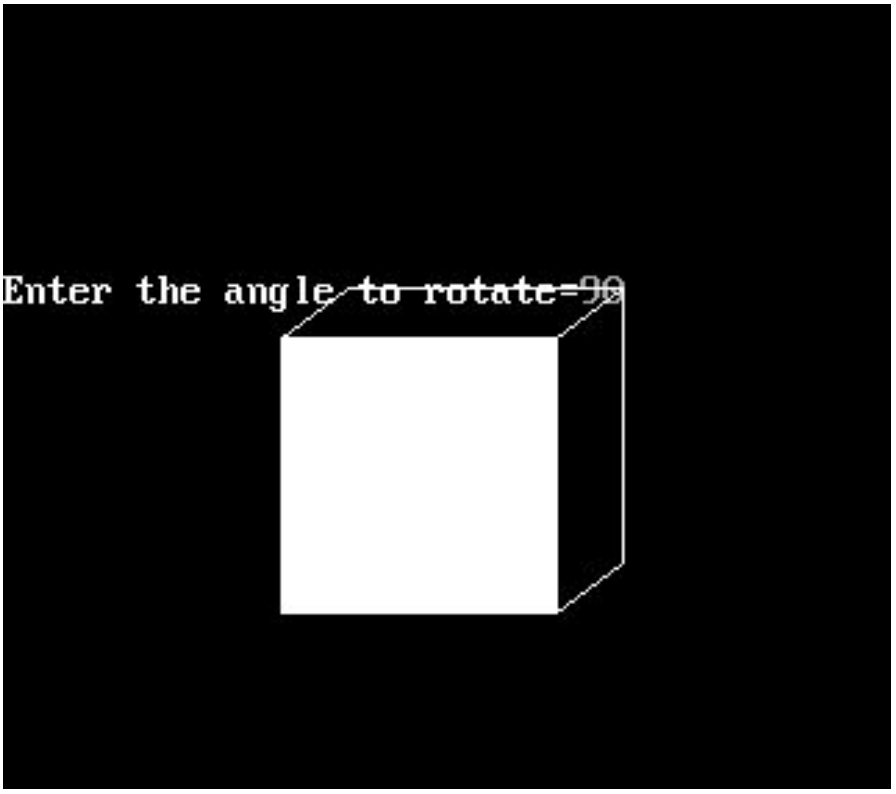
Code:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
int x1,x2,y1,y2,mx,my,depth;
void draw();
void rotate();
void main()
{
  int gd=DETECT,gm,c;
  initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
  printf("\n3D Transformation Rotating\n\n");
  printf("\nEnter 1st top value(x1,y1):");
  scanf("%d%d",&x1,&y1);
  printf("Enter right bottom value(x2,y2):");
  scanf("%d%d",&x2,&y2);
  depth=(x2-x1)/4;
  mx=(x1+x2)/2;
  my=(y1+y2)/2;
  draw(); getch();
  cleardevice();
  rotate();
  getch();
}
void draw()
```

```
{
    bar3d(x1,y1,x2,y2,depth,1);
}
void rotate()
{
    float t;
    int a1,b1,a2,b2,dep;
    printf("Enter the angle to rotate=");
    scanf("%f",&t);
    t=t*(3.14/180);
    a1=mx+(x1-mx)*cos(t)-(y1-my)*sin(t);
    a2=mx+(x2-mx)*cos(t)-(y2-my)*sin(t);
    b1=my+(x1-mx)*sin(t)-(y1-my)*cos(t);
    b2=my+(x2-mx)*sin(t)-(y2-my)*cos(t);
    if(a2>a1)
       dep=(a2-a1)/4;
   else
       dep=(a1-a2)/4;
   bar3d(a1,b1,a2,b2,dep,1); setcolor(5);
}
```

Output:




Enter the angle to rotate=90