# RAM LAL ANAND COLLEGE

# UNIVERSITY OF DELHI

## Department of Computer Science

## Session:- January to May 2021

# Machine Learning Practical File

**Submitted to Sakshi Taaresh Khanna**

Program Name: **B.Sc(H) Computer Science**

Semester: **6**

Title of the paper: **Computer Graphics**

Name of the Student: Arushi

Examination Roll no: 19058570006

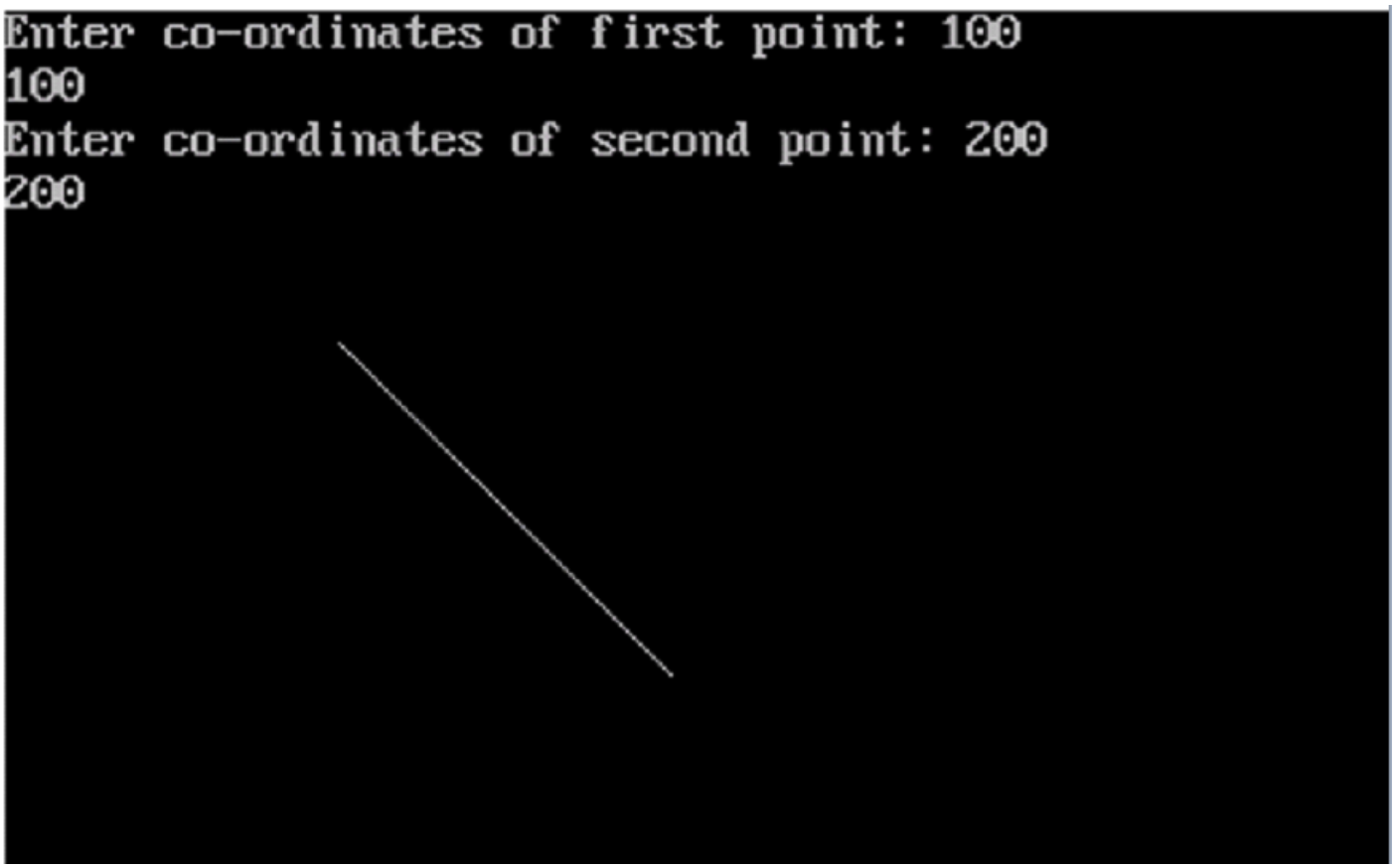
1. Write a program to implement DDA and Bresenham's line drawing algorithm.

Bresenham's line drawing algorithm.

```
#include<graphics.h>

#include<stdio.h>

void main()

{

int x,y,x1,y1,delx,dely,m,grtr_d,smlr_d,d;

int gm,gd=DETECT;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

printf("******* BRESENHAM'S LINE DRAWING ALGORITHM

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

printf("enter initial coordinate = ");

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

printf("enter final coordinate = ");

scanf("%d %d",&x1,&y1);
```

```
delx=x1-x;

dely=y1-y;

grtr_d=2*dely-2*delx; // when d > 0

smlr_d=2*dely;           // when d< 0

d=(2*dely)-delx;



do{

putpixel(x,y,1);

if(d<0) {

d=smlr_d+d;

}

else

{

d=grtr_d+d;

y=y+1;

}

x=x+1;

}while(x<x1);

getch();
```

```
}
```

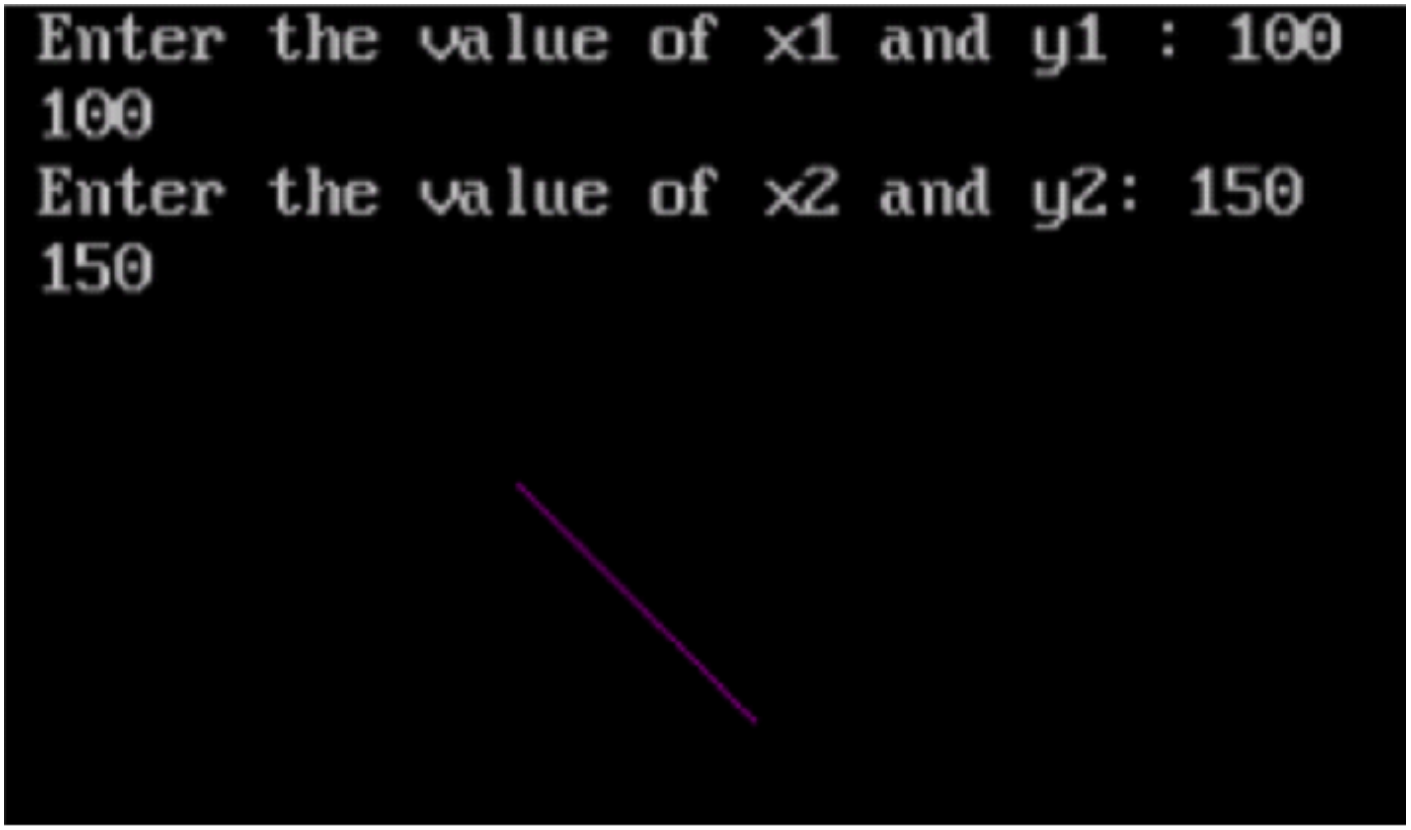



DDA

```
#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(500);
```

```
        }


        closegraph();
}
```

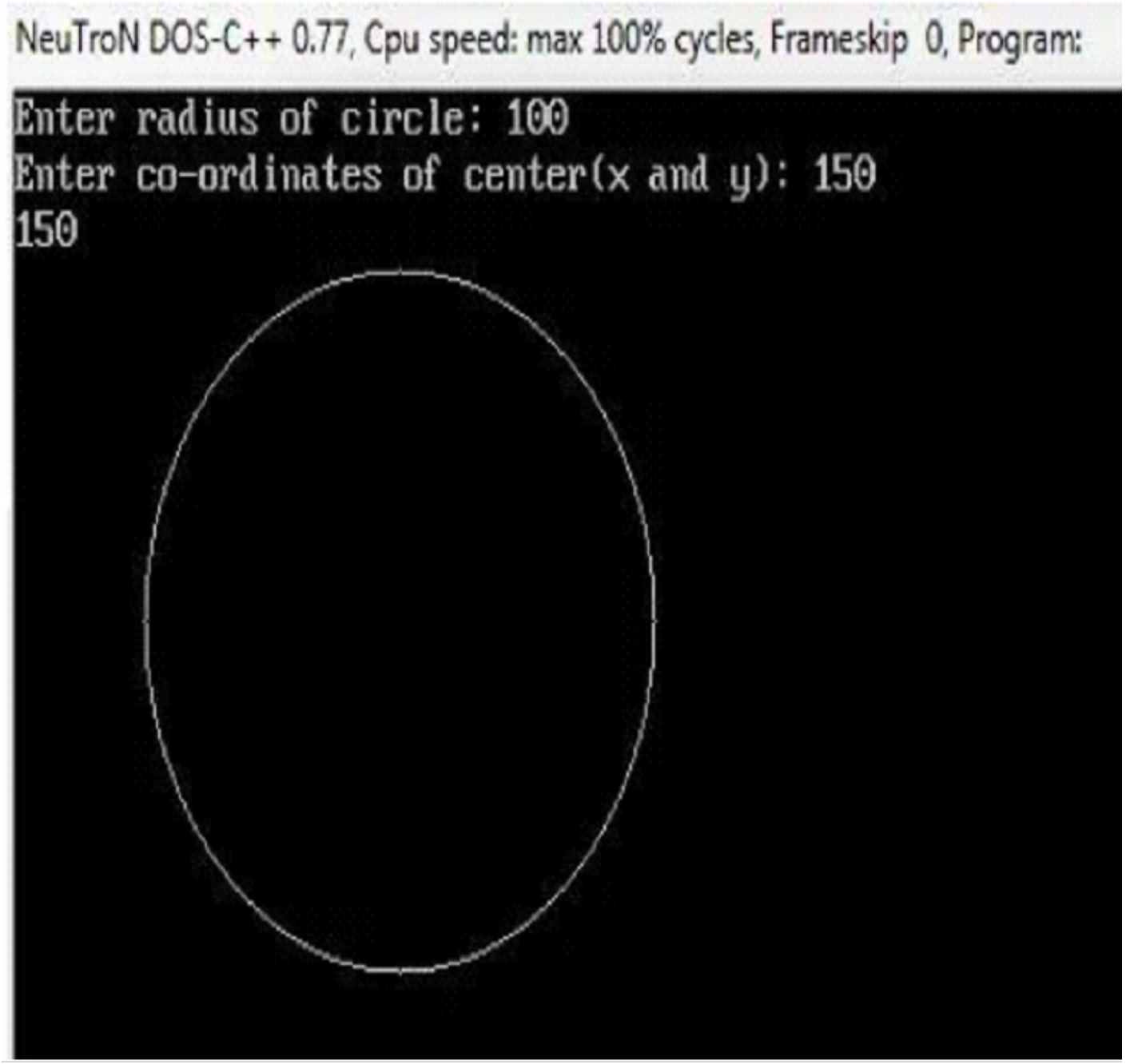



2. Write a program to implement mid-point circle drawing algorithm.

```
#include<iostream.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);

return 0;

}
```




3. Write a program to clip a line using Cohen and Sutherland line clipping algorithm.

```
#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) {
```

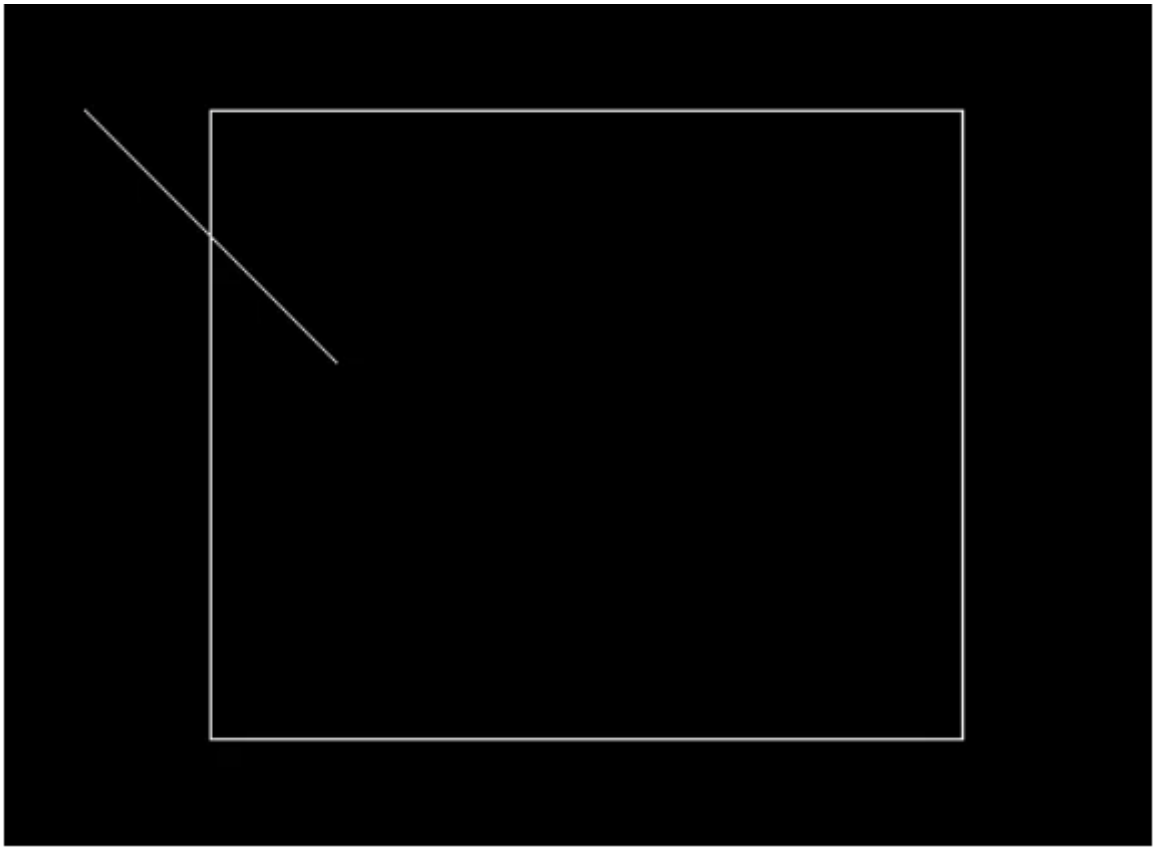
```
        line(p1.x,p1.y,p2.x,p2.y);
    } PT setcode(PT p) //for setting the 4 bit code { PT
    ptemp;
    if(p.y<100)
        ptemp.code[0]='1'; //Top else ptemp.code[0]='0';
    if(p.y>350)
        ptemp.code[1]='1'; //Bottom else ptemp.code[1]='0';
    if(p.x>450)
        ptemp.code[2]='1'; //Right 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;
    }
```

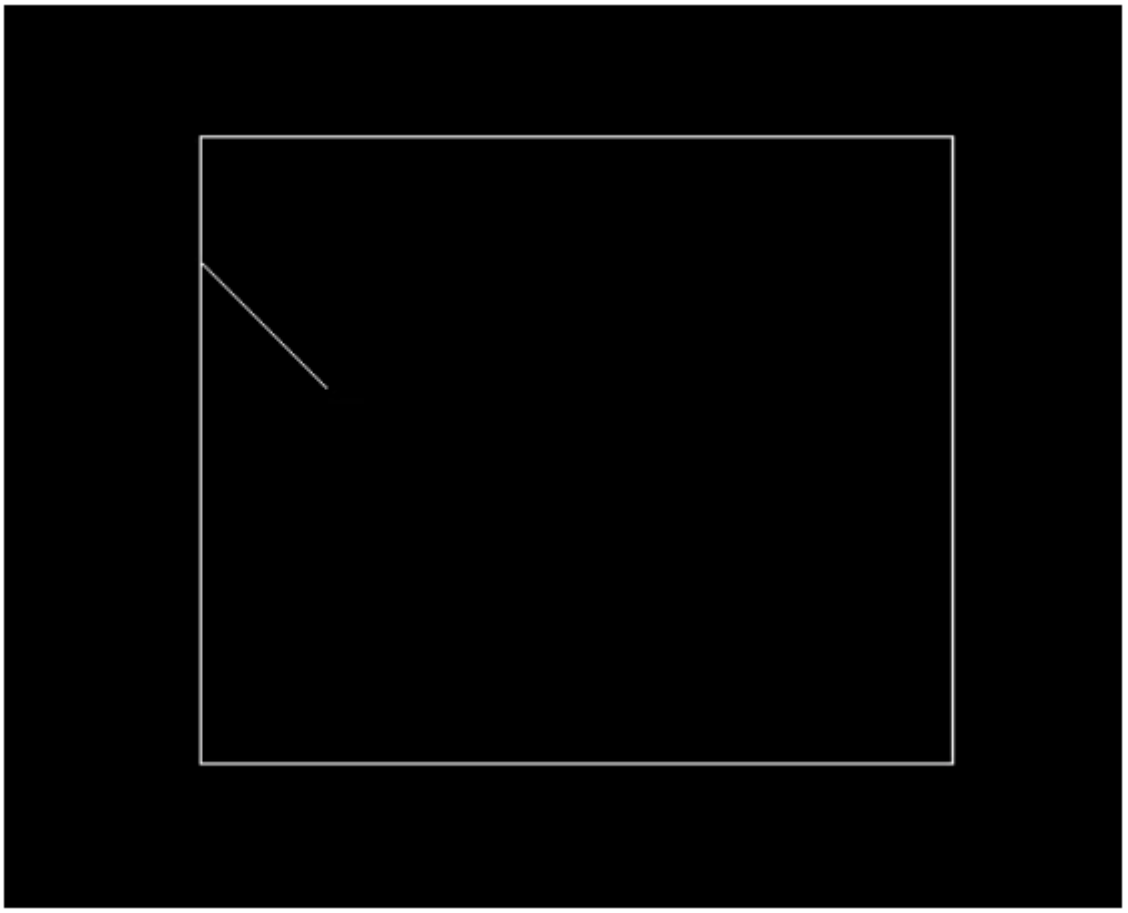
```
    if(flag==0) return(0);
    for(i=0;i<4;i++) {
         if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))
            flag='0';
    }
    if(flag==0)
        return(1);
    return(2);
}
PT resetendpt(PT p1,PT p2) {
    PT temp;
    int x,y,i;
    float m,k;
    if(p1.code[3]=='1') x=150;
    if(p1.code[2]=='1') 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:

```
Enter x1 and y1
100
100

Enter x2 and y2
200
200_
```





4. Write a program to clip a polygon using Sutherland Hodgeman algorithm.

```
#include <stdio.h>

#include <graphics.h>

#include <conio.h>

#include <math.h>

#include <process.h>

#define TRUE 1

#define FALSE 0

typedef unsigned int outcode;
```

```
outcode CompOutCode(float x,float y);

enum { TOP = 0x1,

BOTTOM = 0x2,

RIGHT = 0x4,

LEFT = 0x8};

float xmin,xmax,ymin,ymax;

void clip(float x0,float y0,float x1,float y1){

outcode outcode0,outcode1,outcodeOut;

int accept = FALSE,done = FALSE;

outcode0 = CompOutCode(x0,y0);

outcode1 = CompOutCode(x1,y1);

do{

if(!(outcode0|outcode1)){

accept = TRUE;

done = TRUE;}

else

if(outcode0 & outcode1)

done = TRUE;

else{
```

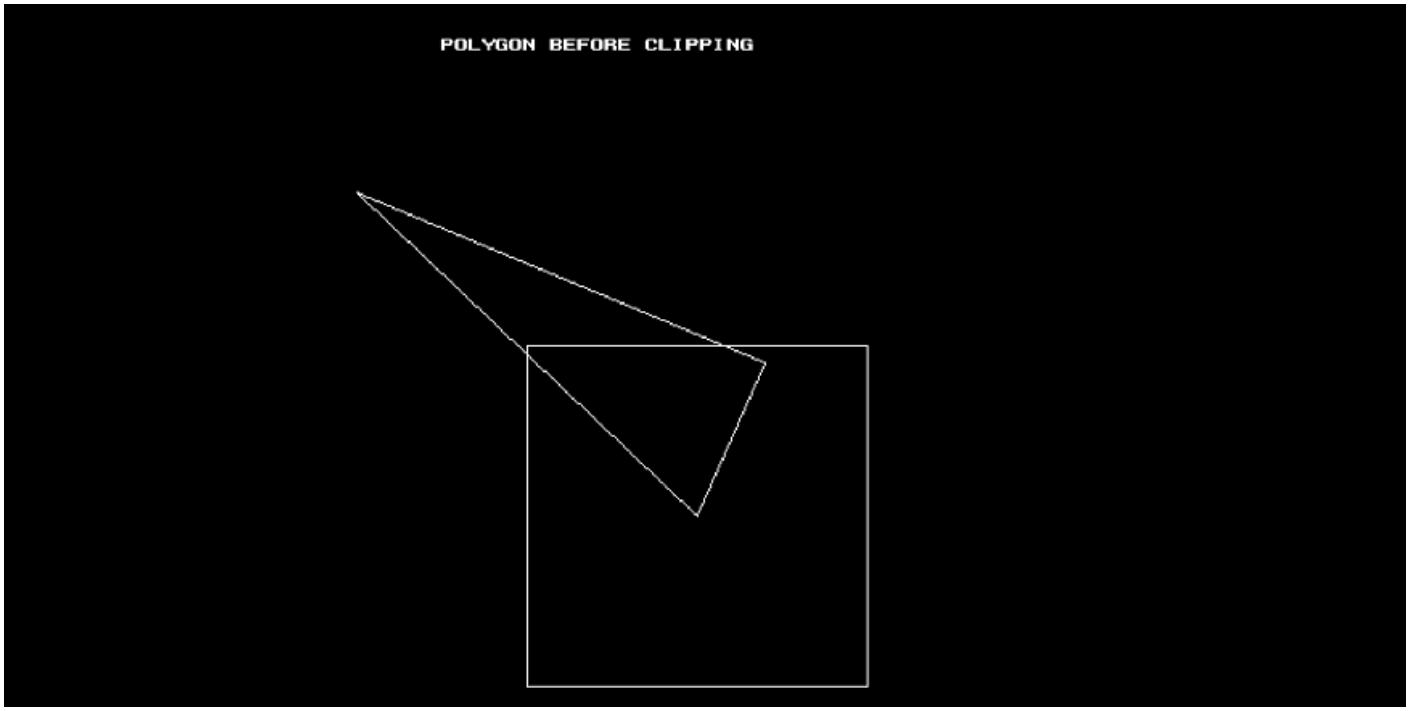
```
float x,y;

outcodeOut = outcode0?outcode0:outcode1;

if(outcodeOut & TOP){

x = x0+(x1-x0)*(ymax-y0)/(y1-y0);

y = ymax;}

else

if(outcodeOut & BOTTOM){

x = x0+(x1-x0)*(ymin-y0)/(y1-y0);

y = ymin;}

else

if(outcodeOut & RIGHT){

y = y0+(y1-y0)*(xmax-x0)/(x1-x0);

x = xmax;}

else{

y = y0+(y1-y0)*(xmin-x0)/(x1-x0);

x = xmin;}

if(outcodeOut==outcode0){

x0 = x;

y0 = y;
```

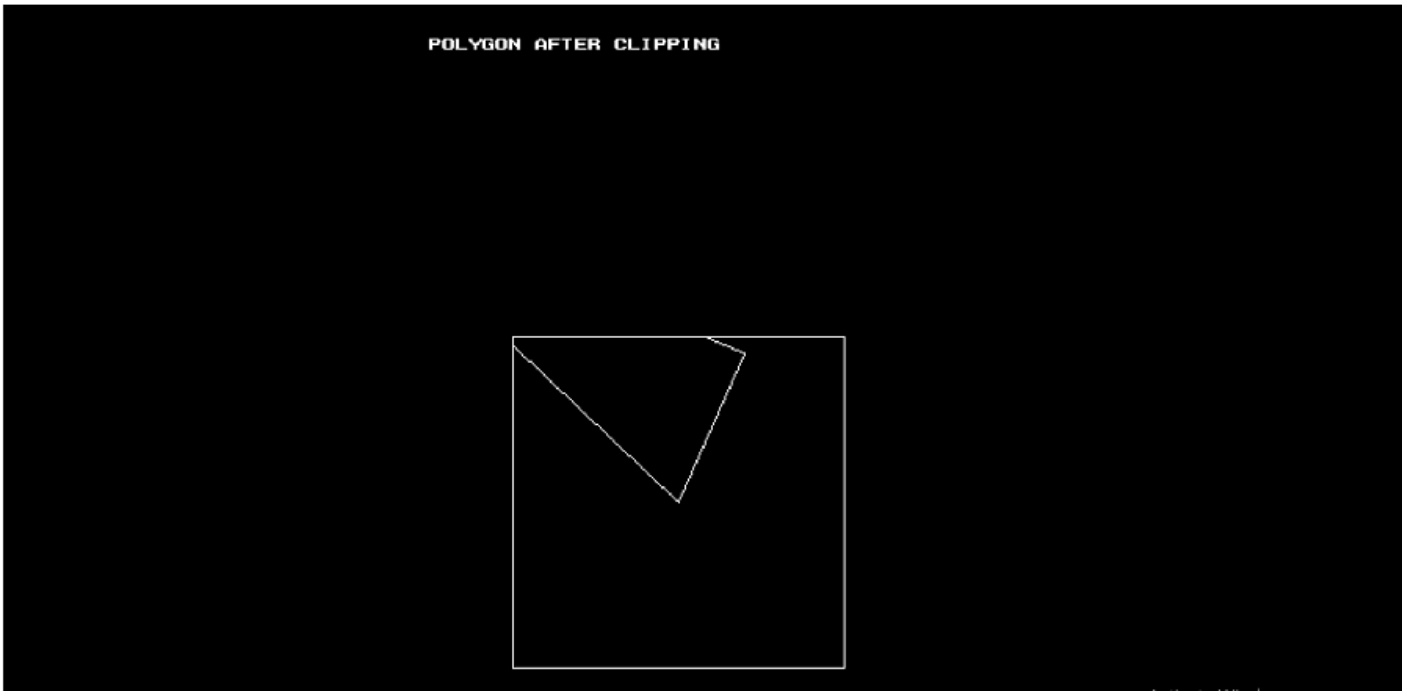
```
outcode0 = CompOutCode(x0,y0);}

else{

x1 = x;

y1 = y;

outcode1 = CompOutCode(x1,y1);}}

}while(done==FALSE);

if(accept)

line(x0,y0,x1,y1);

outtextxy(150,20,”POLYGON AFTER CLIPPING”);

rectangle(xmin,ymin,xmax,ymax);}

outcode CompOutCode(float x,float y){

outcode code = 0;

if(y>ymax)

code|=TOP;

else

if(y<ymin)

code|=BOTTOM;

if(x>xmax)

code|=RIGHT;
```

```
else

if(x<xmin)

code|=LEFT;

return code;}

int main( ){

float x1,y1,x2,y2;

/* request auto detection */

int gdriver = DETECT, gmode, n,poly[14],i;

printf(“Enter the no of sides of polygon:”);

scanf(“%d”,&n);

printf(“\nEnter the coordinates of polygon\n”);

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

{

scanf(“%d”,&poly[i]);

}

poly[2*n]=poly[0];

poly[2*n+1]=poly[1];

printf(“Enter the rectangular coordinates of clipping

window\n”);
```

```
scanf(“%f%f%f%f”,&xmin,&ymin,&xmax,&ymax);

/* initialize graphics and local variables */

initgraph(&gdriver, &gmode, “C://TURBOC3//BGI”);

outtextxy(150,20,”POLYGON BEFORE CLIPPING”);

drawpoly(n+1,poly);

rectangle(xmin,ymin,xmax,ymax);

getch( );

cleardevice( );

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

clip(poly[2*i],poly[(2*i)+1],poly[(2*i)+2],poly[(2*i)+3]);

getch( );

restorecrtmode( );

return 0;

}
```

```
Enter the no of sides of polygon:3
nEnter the coordinates of polygon
100 110
340 210
300 300
Enter the rectangular coordinates of clipping window
200 200 400 400
```






5. Write a program to fill a polygon using Scan line fill algorithm.

```
#include <stdio.h>

#include <conio.h>

#include <graphics.h>

void main()

{

int n,i,j,k,gd,gm,dy,dx;

int x,y,temp;

int a[20][2],xi[20];

float slope[20];

clrscr();
```

```
printf("\n\n\tEnter the no. of edges of polygon : ");

scanf("%d",&n);

printf("\n\n\tEnter the cordinates of polygon :\n\n\n ");

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

{

printf("\tX%d Y%d : ",i,i);

scanf("%d %d",&a[i][0],&a[i][1]);

}

a[n][0]=a[0][0];

a[n][1]=a[0][1];

detectgraph(&gd,&gm);

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

/*- draw polygon -*/

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

{

line(a[i][0],a[i][1],a[i+1][0],a[i+1][1]);

}

getch();

for(i=0;i<n;i++)
```

```
{
dy=a[i+1][1]-a[i][1];
dx=a[i+1][0]-a[i][0];
if(dy==0) slope[i]=1.0;
if(dx==0) slope[i]=0.0;
if((dy!=0)&&(dx!=0)) /*- calculate inverse slope -*/
{
slope[i]=(float) dx/dy;
}
}
for(y=0;y< 480;y++)
{
k=0;
for(i=0;i<n;i++)

 {
if( ((a[i][1]<=y)&&(a[i+1][1]>y))||
((a[i][1]>y)&&(a[i+1][1]<=y)))
{
```

```
xi[k]=(int)(a[i][0]+slope[i]*(y-a[i][1]));

k++;

}

}

for(j=0;j<k-1;j++) /*- Arrange x-intersections in order -*/

for(i=0;i<k-1;i++)

{

if(xi[i]>xi[i+1])

{

temp=xi[i];

xi[i]=xi[i+1];

xi[i+1]=temp;

}

}

setcolor(3);

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

{

line(xi[i],y,xi[i+1]+1,y);

getch();
```

```
}

}

}
```

```
Enter the no. of edges of polygon : 3

Enter the cordinates of polygon :

X0 Y0 : 150 300
X1 Y1 : 300 150
X2 Y2 : 300 300
```

6. Write a program to apply various 2D transformations on a 2D object (use homogenous Coordinates).

```
#include<iostream>

#include<conio.h>

#include<math.h>
```

```
#include<process.h>

#include<graphics.h>

#include<stdlib.h>

using namespace std;


void square(int xa[], int ya[]) {

    line(xa[0], ya[0], xa[1], ya[1]);

    line(xa[1], ya[1], xa[2], ya[2]);

    line(xa[2], ya[2], xa[3], ya[3]);

    line(xa[3], ya[3], xa[0], ya[0]);

}


int main() {

    int
gd=DETECT,gm,n=4,i,xa[10],ya[10],op,tx,ty,xa1[10],ya1[10];

    int theta,rx,ry,sx,sy,shx,shy,xref,yref,axis;

    char d;

    initgraph(&gd, &gm, (char*)"");
```

```
    for(i=0; i<4; i++) {
        cout<<"Enter the coordinates of point "<<i+1<<": \n";
        cout<<"\nEnter the x - coordinate: ";
        cin>>xa[i];
        cout<<"\nEnter the y - coordinate: ";
        cin>>ya[i];
    }

    textattr((BLACK << 8) + WHITE);
    clrscr();

    do {
        cleardevice();
        cout<<"\t\t\t\t\t Menu";

cout<<"\n1.Translation\n2.Rotation\n3.Scaling\n4.Shearing\n5.
Reflection\n6.Exit";
        cout<<"\nEnter your choice: ";
        cin>>op;
```

```
        switch(op) {
            case 1:
                cout<<"\nEnter the x-coordinate of the
translation vector: ";
                cin>>tx;
                cout<<"\nEnter tahe y-coordinate of the
translation vector: ";
                cin>>ty;

                for(int i=0; i<4; i++) {
                    xa1[i]=xa[i]+tx;
                    ya1[i]=ya[i]+ty;
                }

                cleardevice();
                cout<<"Before and after translation:";
                line(320, 0, 320, 430);
                line(0, 240, 640, 240);
```

```
                square(xa, ya);
                getch();
                square(xa1, ya1);
                getch();
                break;

            case 2:
                cout<<"\nEnter the rotation angle: ";
                cin>>theta;
                theta=(theta*3.14)/180;

                for(i=0; i<4; i++) {
                    xa1[i]=abs(320 + (xa[i]-320)*cos(theta) -
(ya[i]-240)*sin(theta));
                    ya1[i]=abs(240 + (xa[i]-320)*sin(theta) +
(ya[i]-240)*cos(theta));
                }

                for(i=0; i<4; i++)
```

```
            cout<<xa[i]<<" "<<ya[i]<<endl;


        for(i=0; i<4; i++)
        cout<<xa1[i]<<" "<<ya1[i]<<endl;


        getch();
        cleardevice();
        cout<<"\nBefore and After rotation: ";
        line(320, 0, 320, 430);
        line(0, 240, 640, 240);
        square(xa, ya);
        getch();
        square(xa1, ya1);
        getch();
        clrscr();
        break;


    case 3:
        cout<<"\nEnter the scaling factor(Sx and
```

```
Sy):\n";
                cin>>sx>>sy;
                cout<<"\nEnter the reference point: \n";
                cin>>rx>>ry;

                for(i=0; i<n; i++) {
                    xa1[i]=xa[i]*sx+rx*(1-sx);
                    ya1[i]=ya[i]*sy+ry*(1-sy);
                }

                cleardevice();
                cout<<"\nBefore and after scaling:\n";
                line(320, 0, 320, 430);
                line(0, 240, 640, 240);
                square(xa, ya);
                getch();
                square(xa1, ya1);
                getch();
                cleardevice();
```

```
                break;


            case 4:
                cout<<"\nEnter the shear value(SHx & SHy):
\n";
                cin>>shx>>shy;
                cout<<"\nEnter the reference point(Xref &
Yref): \n";
                cin>>xref>>yref;


                for(i=0; i<n; i++) {
                    xa1[i]=xa[i]+shx*(ya[i] - yref);
                    ya1[i]=ya[i]+shy*(xa[i] - xref);
                }


                cleardevice();
                cout<<"\nBefore and after shearing:";
                line(320, 0, 320, 430);
                line(0, 240, 640, 240);
```

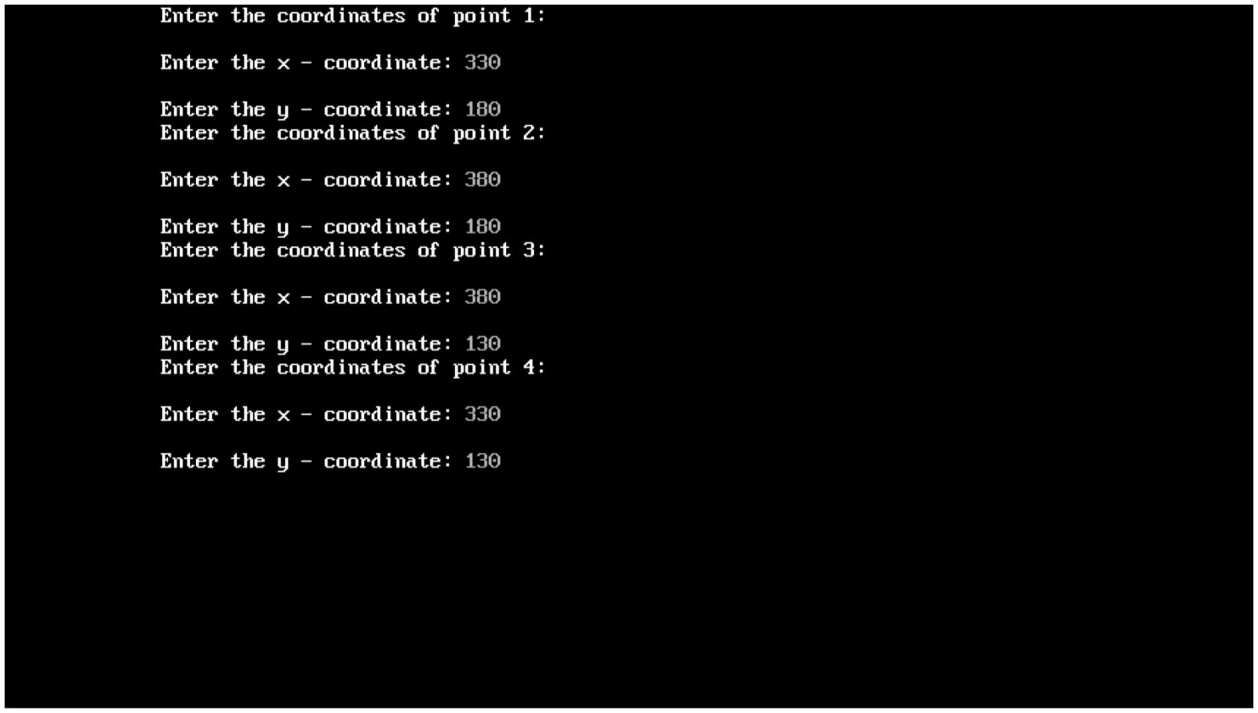
```
                square(xa, ya);
                getch();
                square(xa1, ya1);
                getch();
                cleardevice();
                break;

            case 5:
                cout<<"\nReflect along : \n1. X-Axis\n2.
Y-Axis\n? : ";
                cin>>axis;

                switch(axis) {
                    case 1:
                        for(i=0; i<n; i++) {
                        xa1[i]=xa[i];
                        ya1[i]=480 - ya[i];
                    }
                break;
```

```
        case 2:
            for(i=0; i<n; i++) {
                xa1[i]=640 - xa[i];
                ya1[i]=ya[i];
            }
        break;
    }

        cout<<"\nBefore and after reflection: \n";
        line(320, 0, 320, 430);
        line(0, 240, 640, 240);
        square(xa, ya);
        getch();
        square(xa1, ya1);
        getch();
        cleardevice();
        break;
    case 6:
```

```
                exit(0);

                break;

        }


        cleardevice();

    }

    while(op !=6);

}
```

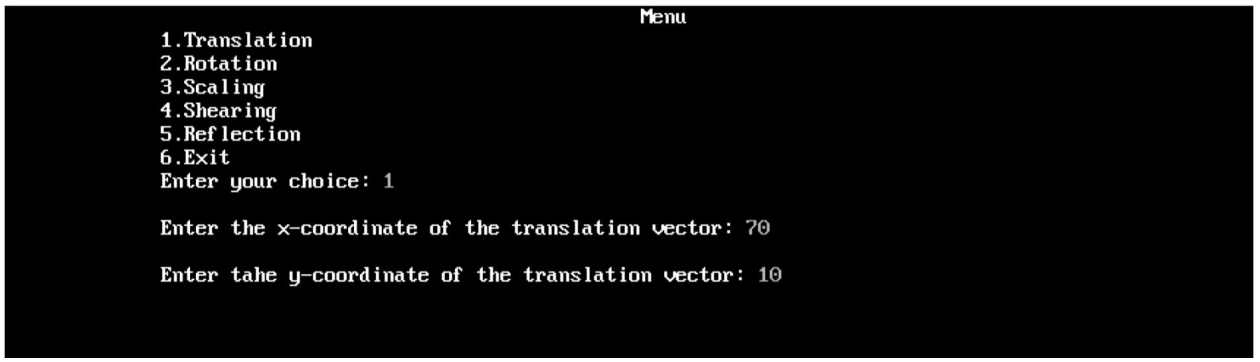
## OUTPUT

Entering the co-ordinates of the Square

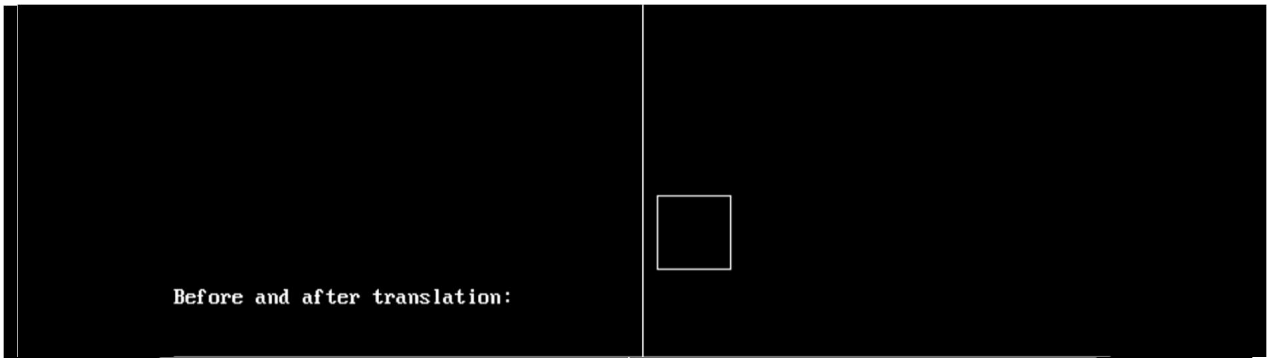



**Translation**

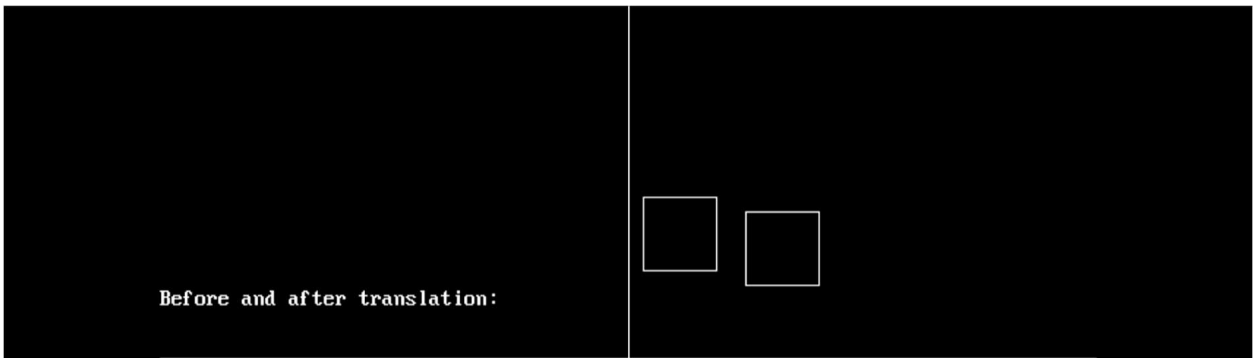
Entering the translation vector




Before Translation

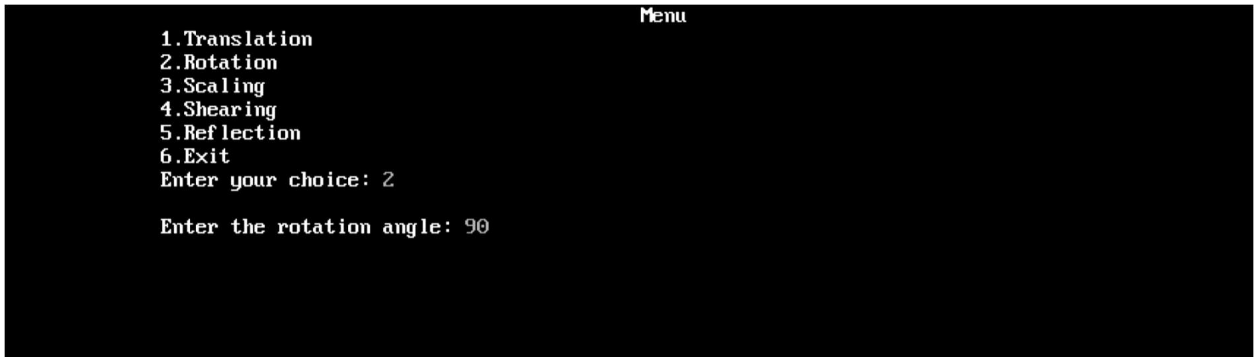



After Translation

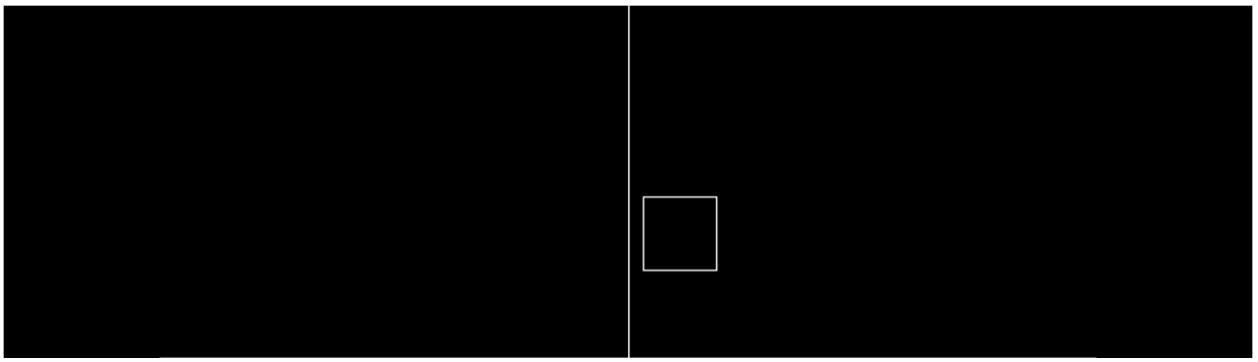



**Rotation**

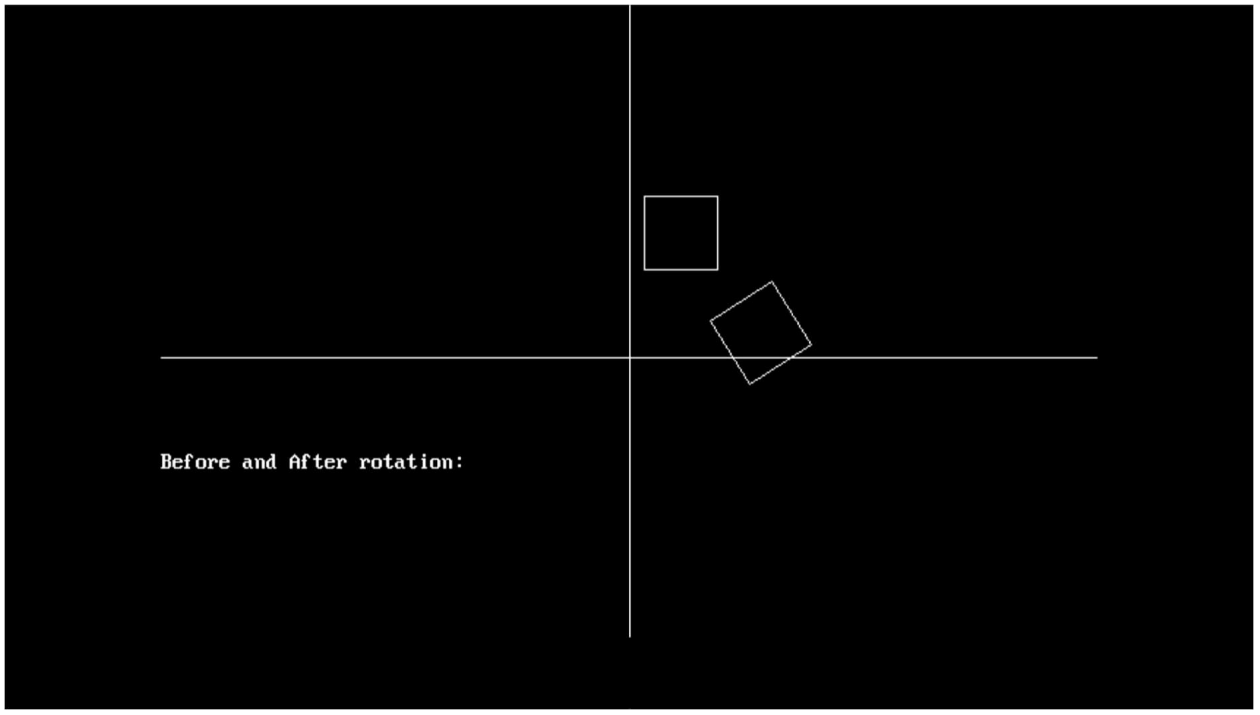
Entering Angle



Before Rotation

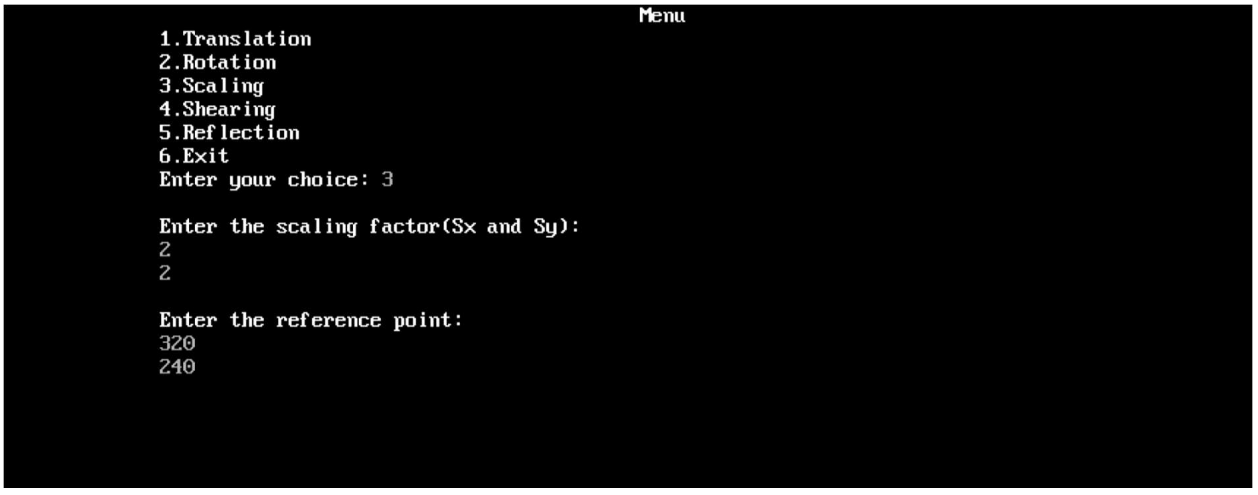


After Rotation

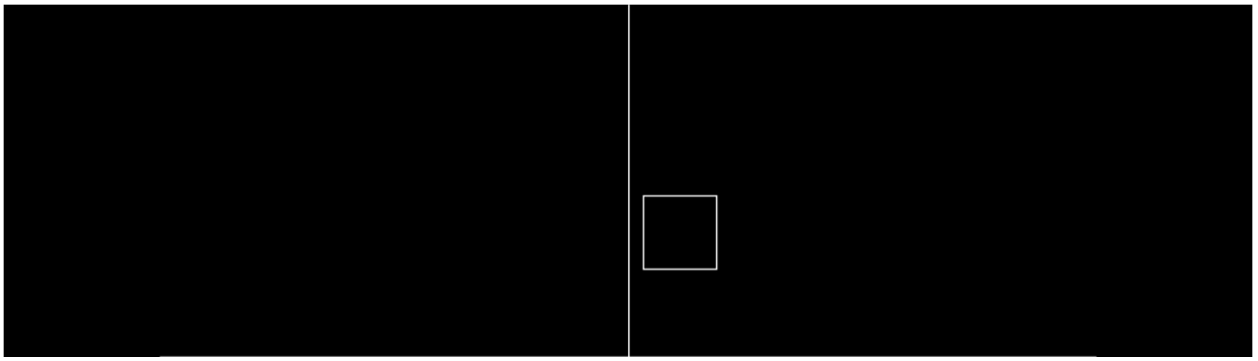



**Scaling**

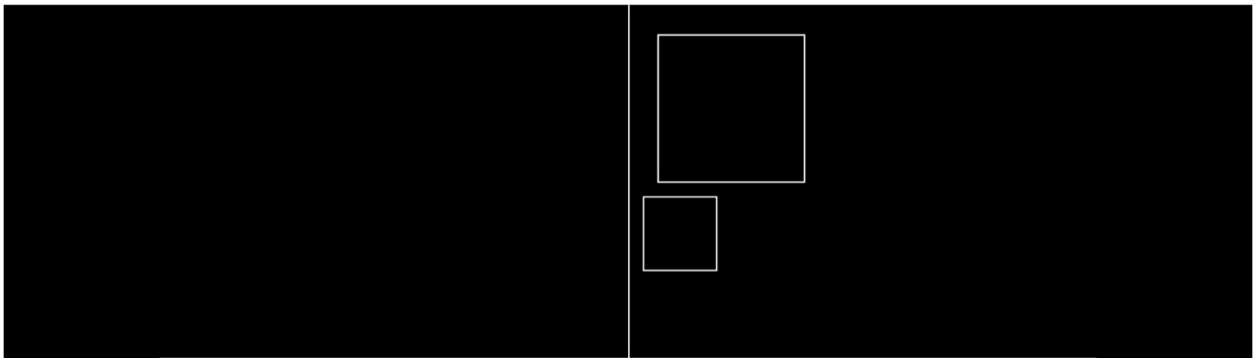
Entering Scaling Factor



Before Scaling



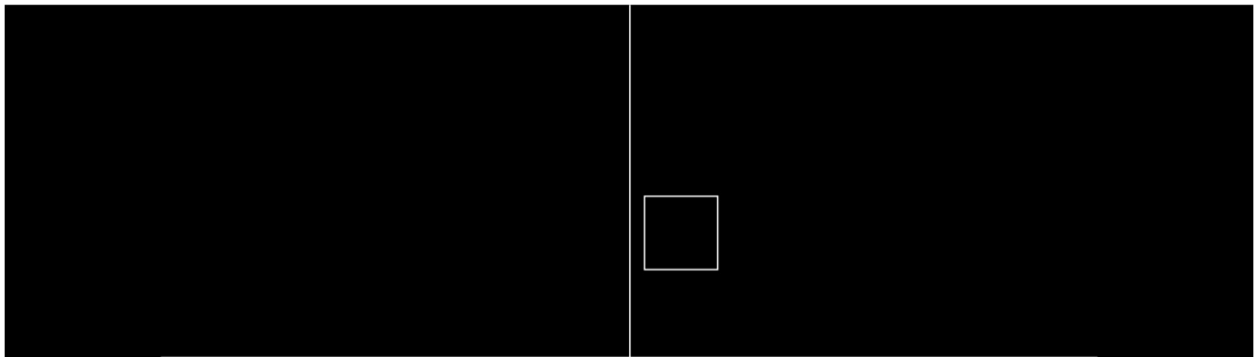
After Scaling



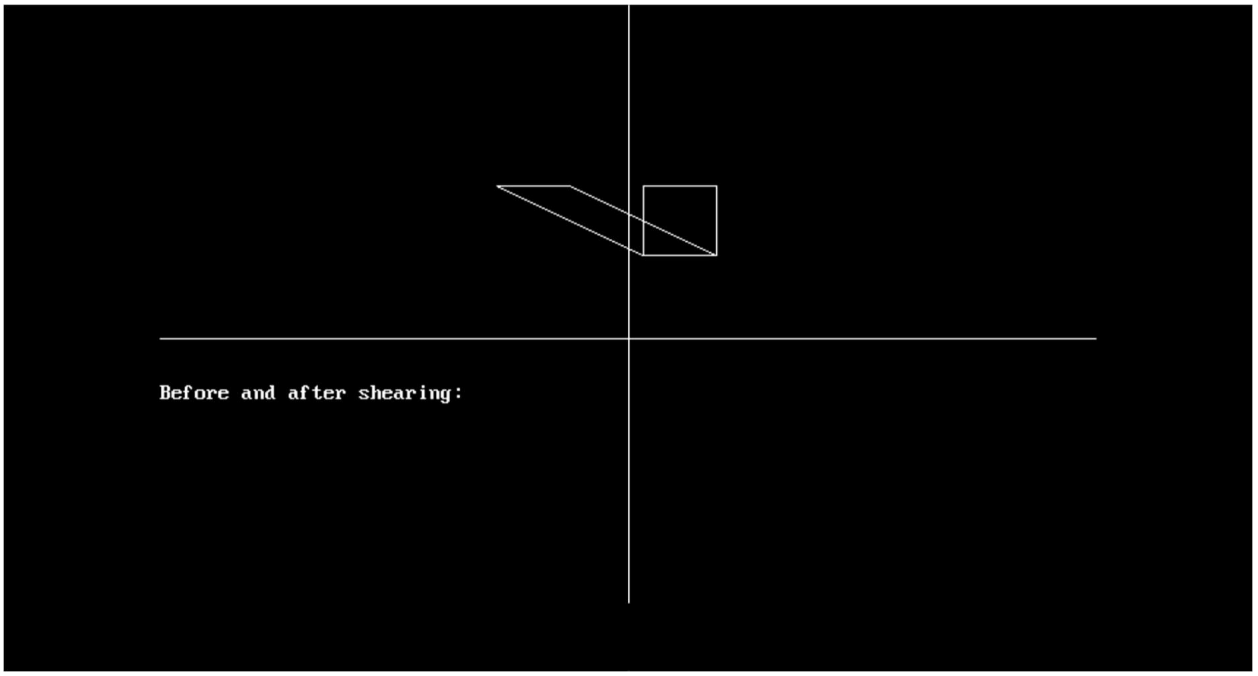
**Shearing**

Entering Shear Value

Before Shearing

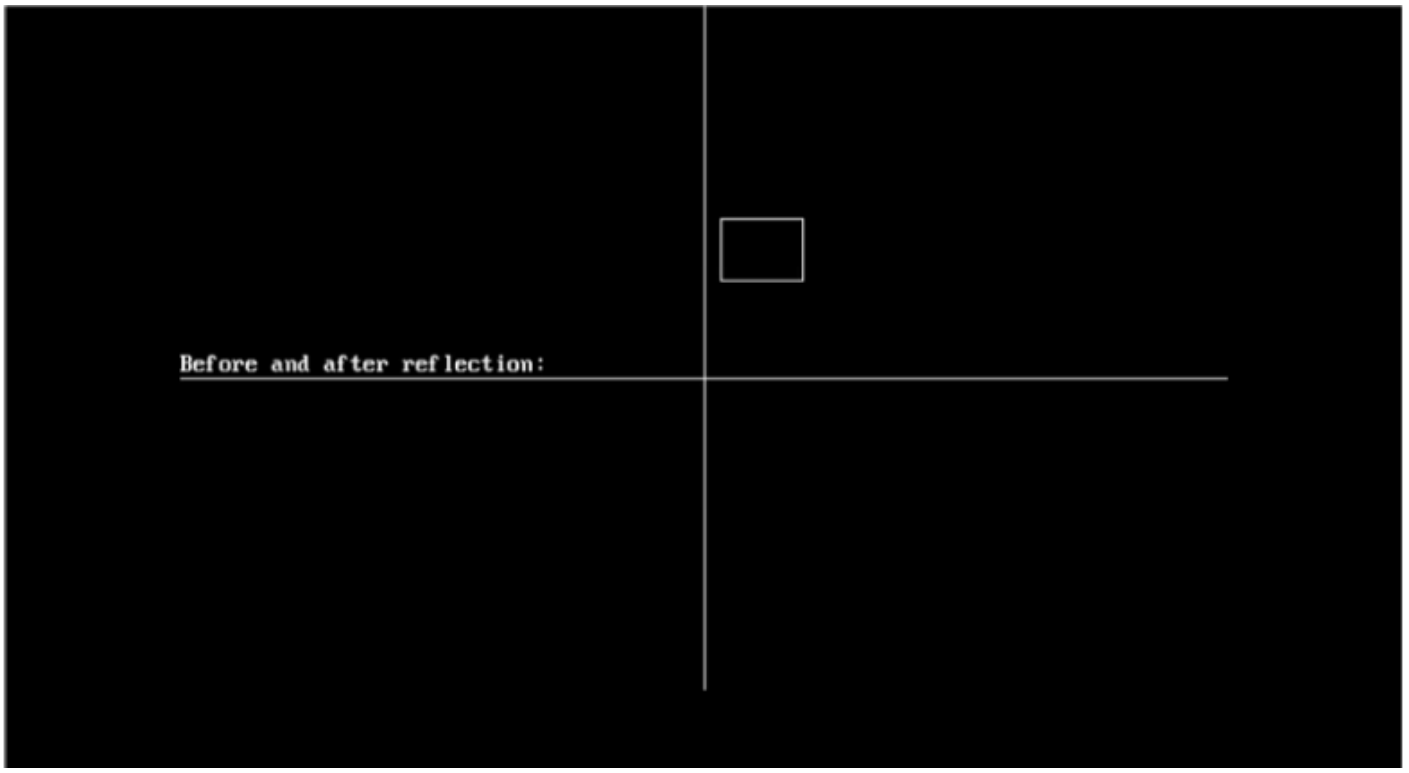


After Shearing

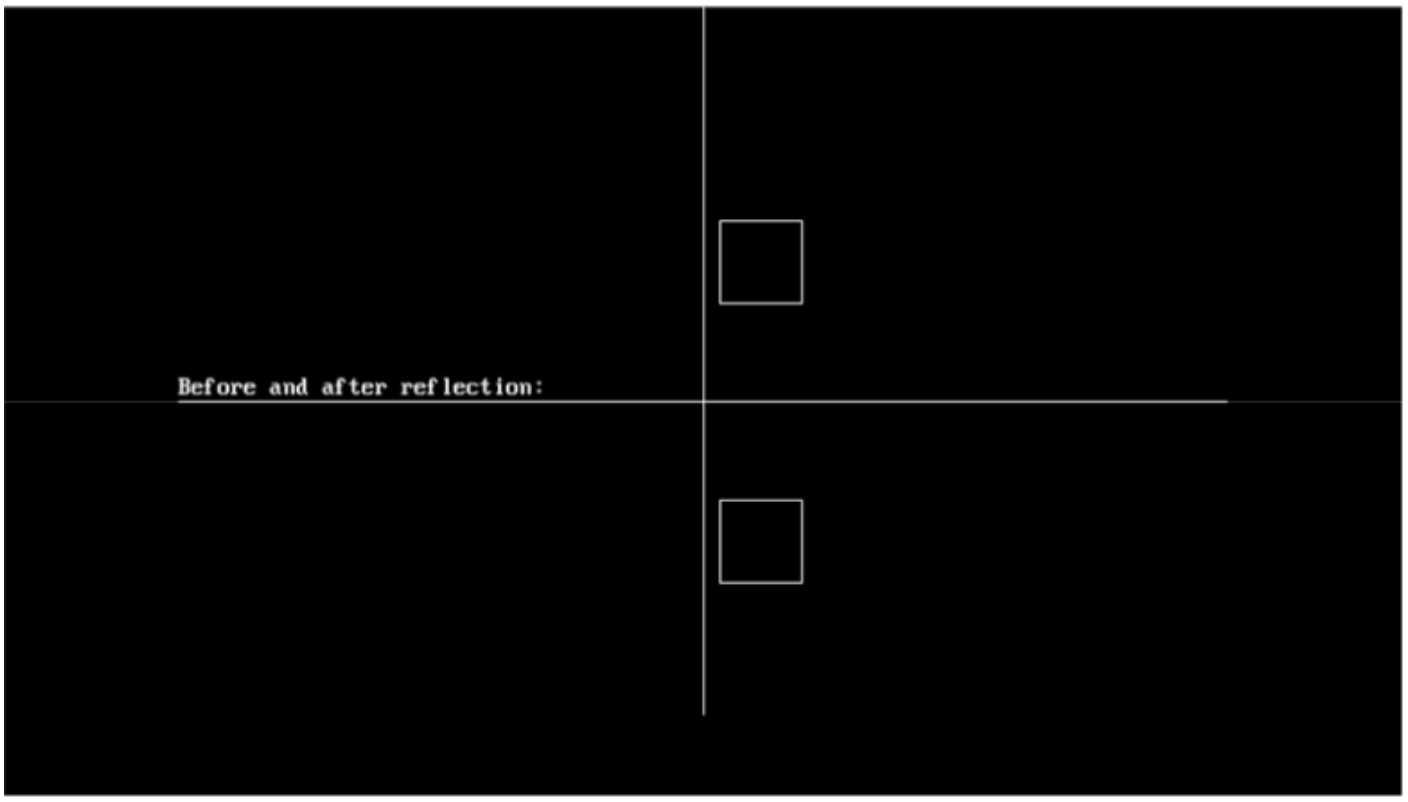



**Reflection**

**Along X-Axis**
Before Reflection

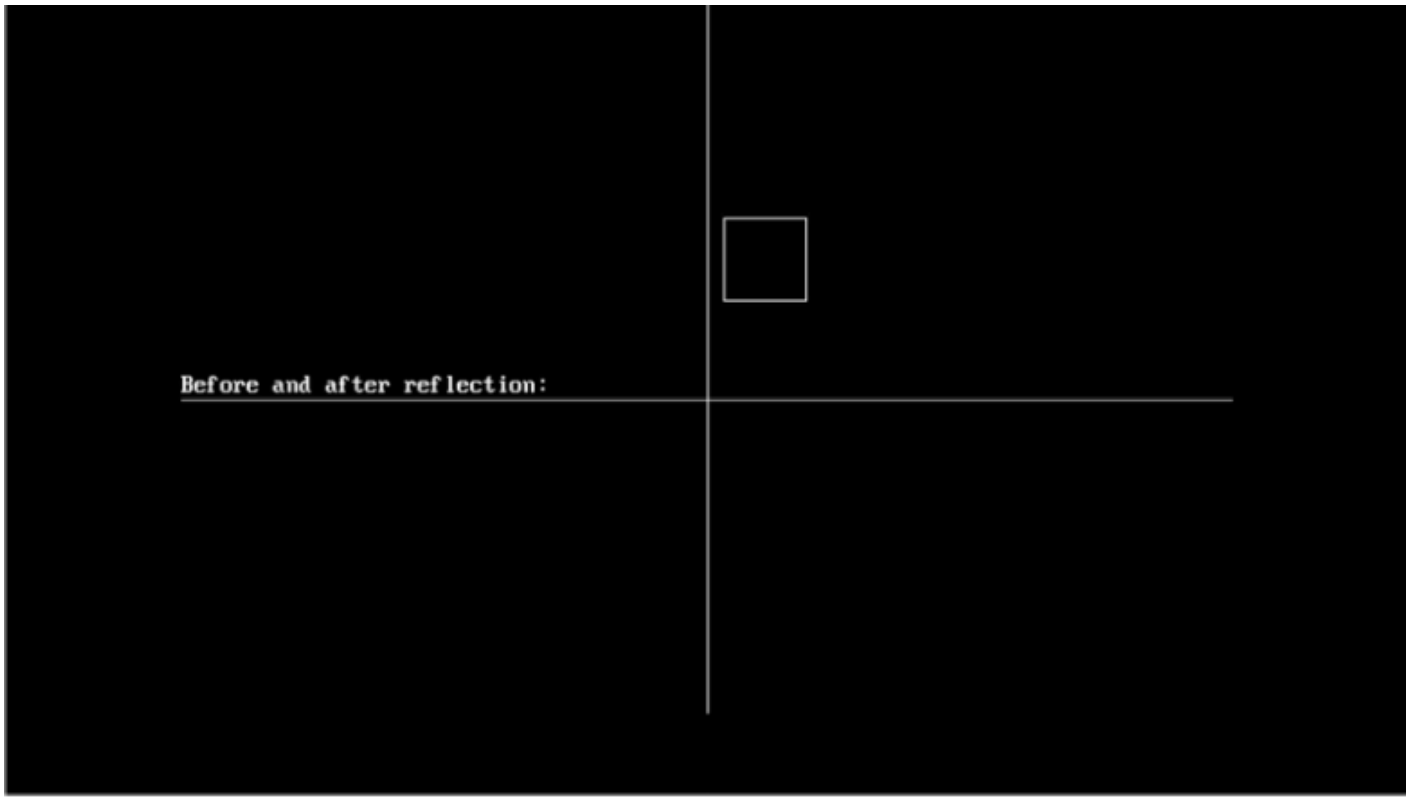



After Reflection

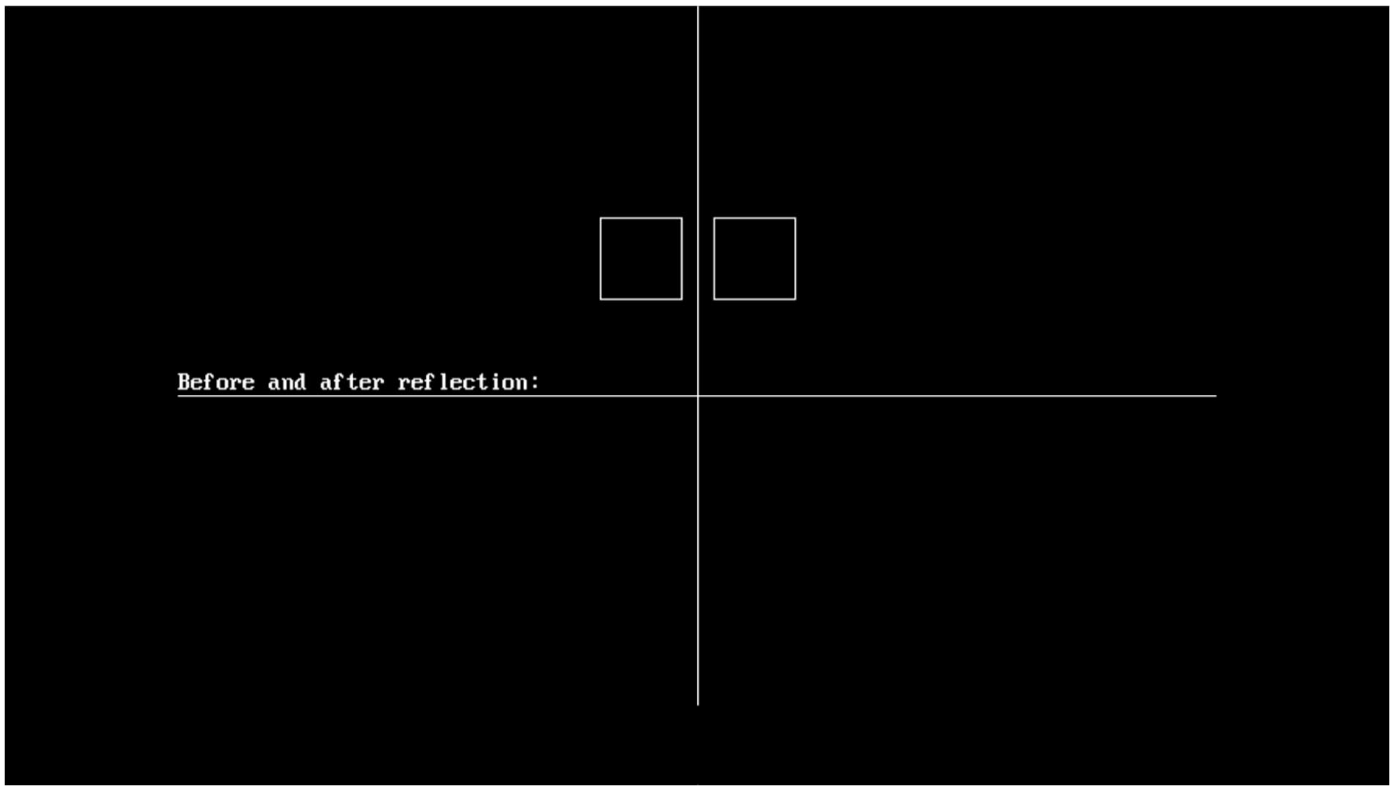



**Along Y-Axis**

Before Reflection




After Reflection



7. Write a program to apply various 3D transformations on a 3D object and then apply parallel and perspective projection on it.

```
#include<iostream.h>

#include<dos.h>

#include<stdio.h>

#include<math.h>

#include<conio.h>

#include<graphics.h>

#include<process.h>



int gd=DETECT,gm;

double x1,x2,y1,y2;



    void draw_cube(double edge[20][3])

    {
```

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

int i;

clearviewport();

for(i=0;i < 19;i++)

{

    x1=edge[i][0]+edge[i][2]*(cos(2.3562));

    y1=edge[i][1]-edge[i][2]*(sin(2.3562));

    x2=edge[i+1][0]+edge[i+1][2]*(cos(2.3562));

    y2=edge[i+1][1]-edge[i+1][2]*(sin(2.3562));

    line(x1+320,240-y1,x2+320,240-y2);

}

line(320,240,320,25);

line(320,240,550,240);

line(320,240,150,410);

getch();

closegraph();

}



 void scale(double edge[20][3])
```

```
{
double a,b,c;
int i;
cout<<"Enter The Scaling Factors "<<endl;
cin>>a>>b>>c;
initgraph(&gd,&gm,"C://TURBOC3//BGI");
clearviewport();
for(i=0;i < 20;i++)
{
    edge[i][0]=edge[i][0]*a;
    edge[i][1]=edge[i][1]*b;
    edge[i][2]=edge[i][2]*c;
}
draw_cube(edge);
closegraph();
 }

 void translate(double edge[20][3])
{
```

```
    int a,b,c;
    int i;
cout<<"Enter The Translation Factors"<<endl;
cin>>a>>b>>c;
initgraph(&gd,&gm,"C://TURBOC3//BGI");
clearviewport();
    for(i=0;i < 20;i++)
{
    edge[i][0]+=a;
    edge[i][0]+=b;
    edge[i][0]+=c;
}
draw_cube(edge);
    closegraph();
}


 void rotate(double edge[20][3])
{
    int ch;
```

```
    int i;
    double temp,theta,temp1;
    clrscr();
cout<<"-=[ Rotation About ]=-"<<endl;
cout<<"1:==> X-Axis "<<endl;
cout<<"2:==> Y-Axis"<<endl;
cout<<"3:==> Z-Axis "<<endl;
cout<<"Enter Your Choice "<<endl;
cin>>ch;
switch(ch)
    {
        case 1:
        cout<<" Enter The Angle ";
        cin>>theta;
        theta=(theta*3.14)/180;
        for(i=0;i < 20;i++)
        {
            edge[i][0]=edge[i][0];
            temp=edge[i][1];
```

```
    temp1=edge[i][2];
    edge[i][1]=temp*cos(theta)-temp1*sin(theta);
    edge[i][2]=temp*sin(theta)+temp1*cos(theta);
}
draw_cube(edge);
break;

case 2:
cout<<" Enter The Angle ";
cin>>theta;
theta=(theta*3.14)/180;
for(i=0;i < 20;i++)
{
    edge[i][1]=edge[i][1];
    temp=edge[i][0];
    temp1=edge[i][2];
    edge[i][0]=temp*cos(theta)+temp1*sin(theta);
    edge[i][2]=-temp*sin(theta)+temp1*cos(theta);
}
```

```
            draw_cube(edge);
            break;

        case 3:
            cout<<" Enter The Angle ";
            cin>>theta;
            theta=(theta*3.14)/180;
            for(i=0;i < 20;i++)
            {
                edge[i][2]=edge[i][2];
                temp=edge[i][0];
                temp1=edge[i][1];
                edge[i][0]=temp*cos(theta)-temp1*sin(theta);
                edge[i][1]=temp*sin(theta)+temp1*cos(theta);
            }
            draw_cube(edge);
            break;
        }
}
```

```
void reflect(double edge[20][3])
{
    int ch;
    int i;
    clrscr();
cout<<"-=[ Reflection About ]=-"<<endl;
cout<<"1:==> X-Axis"<<endl;
cout<<"2:==> Y-Axis "<<endl;
cout<<"3:==> Z-Axis "<<endl;
cout<<" Enter Your Choice "<<endl;
cin>>ch;
switch(ch)
    {
        case 1:
        for(i=0;i < 20;i++)
        {
            edge[i][0]=edge[i][0];
            edge[i][1]=-edge[i][1];
```

```
    edge[i][2]=-edge[i][2];
}
draw_cube(edge);
break;

case 2:
for(i=0;i < 20;i++)
{
    edge[i][1]=edge[i][1];
    edge[i][0]=-edge[i][0];
    edge[i][2]=-edge[i][2];
}
draw_cube(edge);
break;

case 3:
for(i=0;i < 20;i++)
{
    edge[i][2]=edge[i][2];
```

```
            edge[i][0]=-edge[i][0];
            edge[i][1]=-edge[i][1];
        }
        draw_cube(edge);
        break;
    }
}

void perspect(double edge[20][3])
{
    int ch;
    int i;
    double p,q,r;
    clrscr();
cout<<"-=[ Perspective Projection About ]=-"<<endl;
cout<<"1:==> X-Axis "<<endl;
cout<<"2:==> Y-Axis "<<endl;
cout<<"3:==> Z-Axis"<<endl;
cout<<" Enter Your Choice :="<<endl;
```

```
cin>>ch;
switch(ch)
    {
        case 1:
        cout<<" Enter P :=";
        cin>>p;
        for(i=0;i < 20;i++)
        {
            edge[i][0]=edge[i][0]/(p*edge[i][0]+1);
            edge[i][1]=edge[i][1]/(p*edge[i][0]+1);
            edge[i][2]=edge[i][2]/(p*edge[i][0]+1);
        }
        draw_cube(edge);
        break;

    case 2:
        cout<<" Enter Q :=";
        cin>>q;
        for(i=0;i < 20;i++)
```

```
{
    edge[i][1]=edge[i][1]/(edge[i][1]*q+1);
    edge[i][0]=edge[i][0]/(edge[i][1]*q+1);
    edge[i][2]=edge[i][2]/(edge[i][1]*q+1);
}
draw_cube(edge);
break;


case 3:
cout<<" Enter R :=";
cin>>r;
for(i=0;i < 20;i++)
{
    edge[i][2]=edge[i][2]/(edge[i][2]*r+1);
    edge[i][0]=edge[i][0]/(edge[i][2]*r+1);
    edge[i][1]=edge[i][1]/(edge[i][2]*r+1);
}
draw_cube(edge);
break;
```

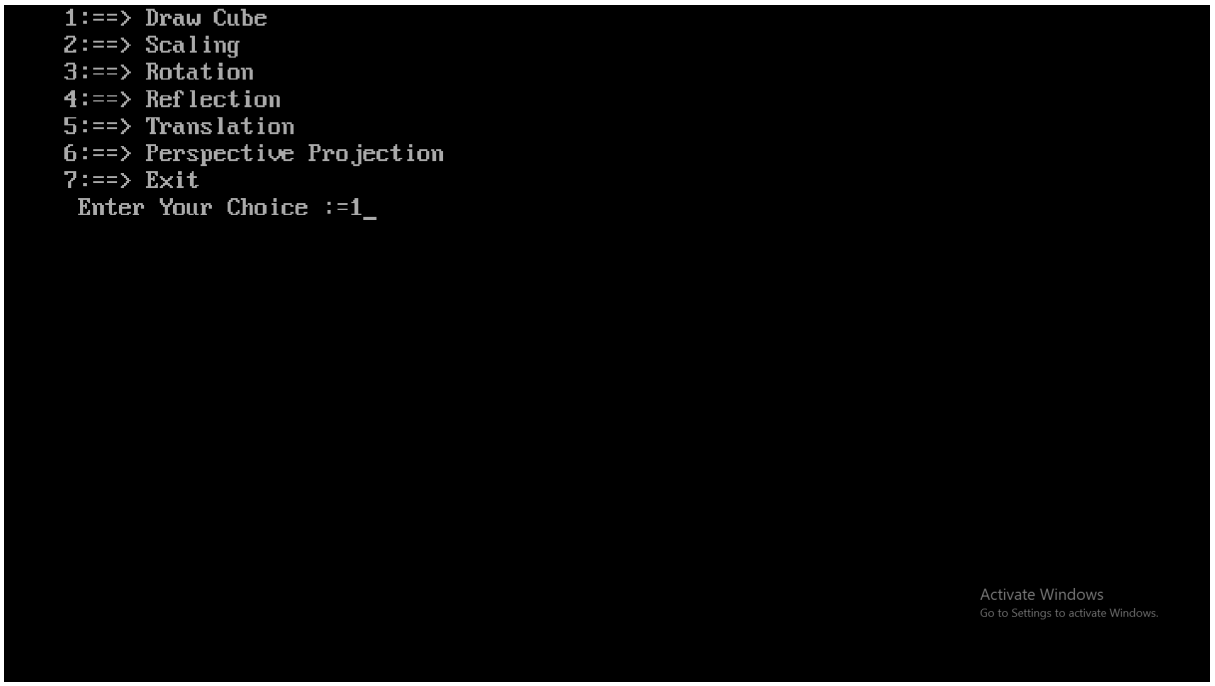
```
        }
        closegraph();
    }

void main()
    {
        int choice;
        double edge[20][3]={
                100,0,0,
                100,100,0,
                0,100,0,
                0,100,100,
                0,0,100,
                0,0,0,
                100,0,0,
                100,0,100,
                100,75,100,
                75,100,100,
                100,100,75,
```

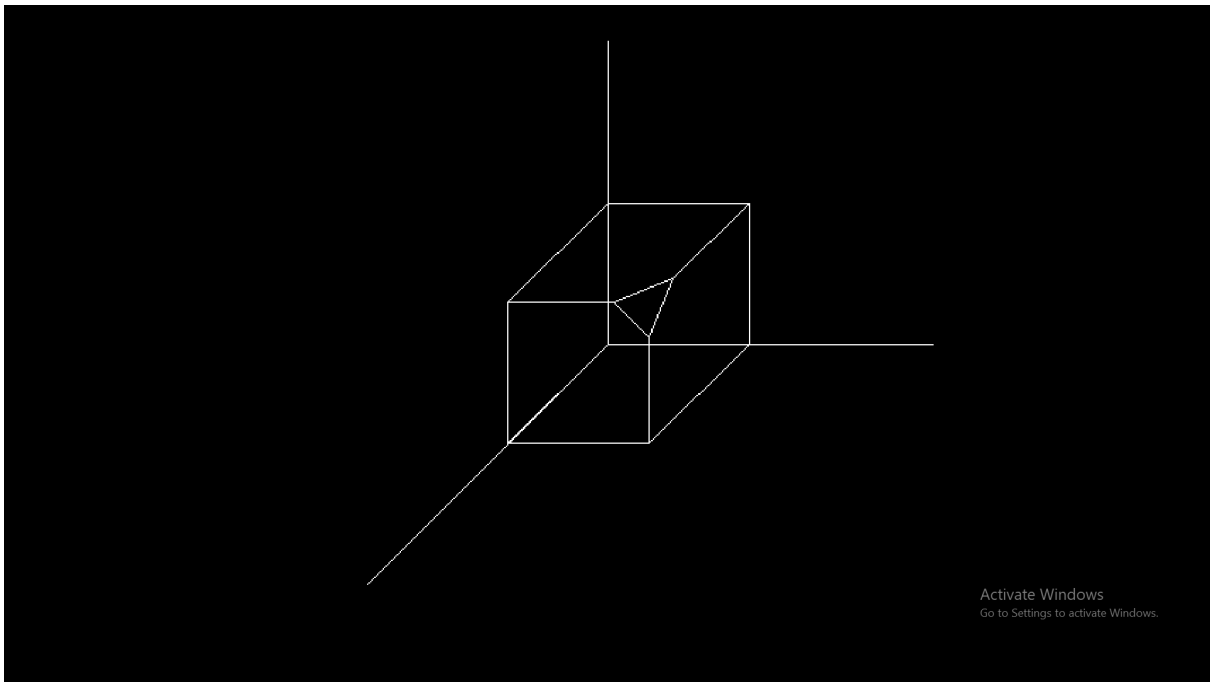
```
          100,100,0,

          100,100,75,

          100,75,100,

          75,100,100,

          0,100,100,

          0,100,0,

          0,0,0,

          0,0,100,

          100,0,100

              };

while(1)

{

clrscr();

//show_message();

cout<<"1:==> Draw Cube "<<endl;

cout<<"2:==> Scaling "<<endl;

cout<<"3:==> Rotation "<<endl;

cout<<"4:==> Reflection "<<endl;

cout<<"5:==> Translation "<<endl;
```

```
cout<<"6:==> Perspective Projection "<<endl;

cout<<"7:==> Exit "<<endl;

cout<<" Enter Your Choice :=";

cin>>choice;

switch(choice)

{

case 1:

    draw_cube(edge);

    break;



case 2:

    scale(edge);

    break;



case 3:

    rotate(edge);

    break;



case 4:
```

```
        reflect(edge);
        break;

    case 5:
        translate(edge);
        break;

    case 6:
        perspect(edge);
        break;

    case 7:
        exit(0);

    default:
        cout<<" Press A Valid Key...!!! ";
        getch();
        break;
    }
```

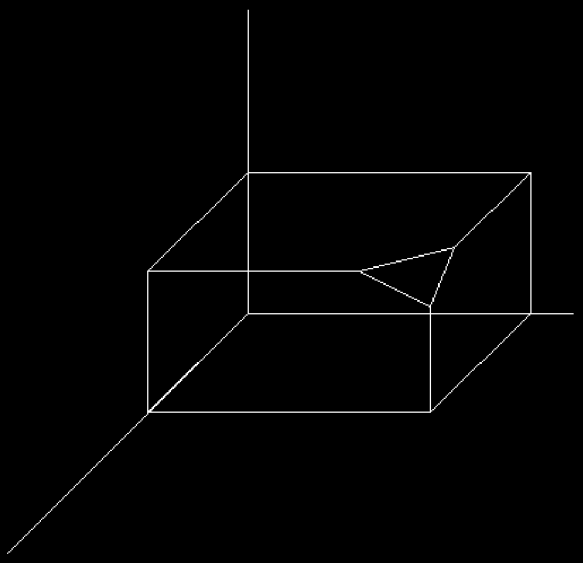
```
    closegraph();

    }

}
```


1:==> Draw Cube
2:==> Scaling
3:==> Rotation
4:==> Reflection
5:==> Translation
6:==> Perspective Projection
7:==> Exit
Enter Your Choice :=1
Activate Windows
Go to Settings to activate Windows.


Activate Windows
Go to Settings to activate Windows.

```
1:==> Draw Cube
2:==> Scaling
3:==> Rotation
4:==> Reflection
5:==> Translation
6:==> Perspective Projection
7:==> Exit
 Enter Your Choice :=2
Enter The Scaling Factors
2 1 1_
```

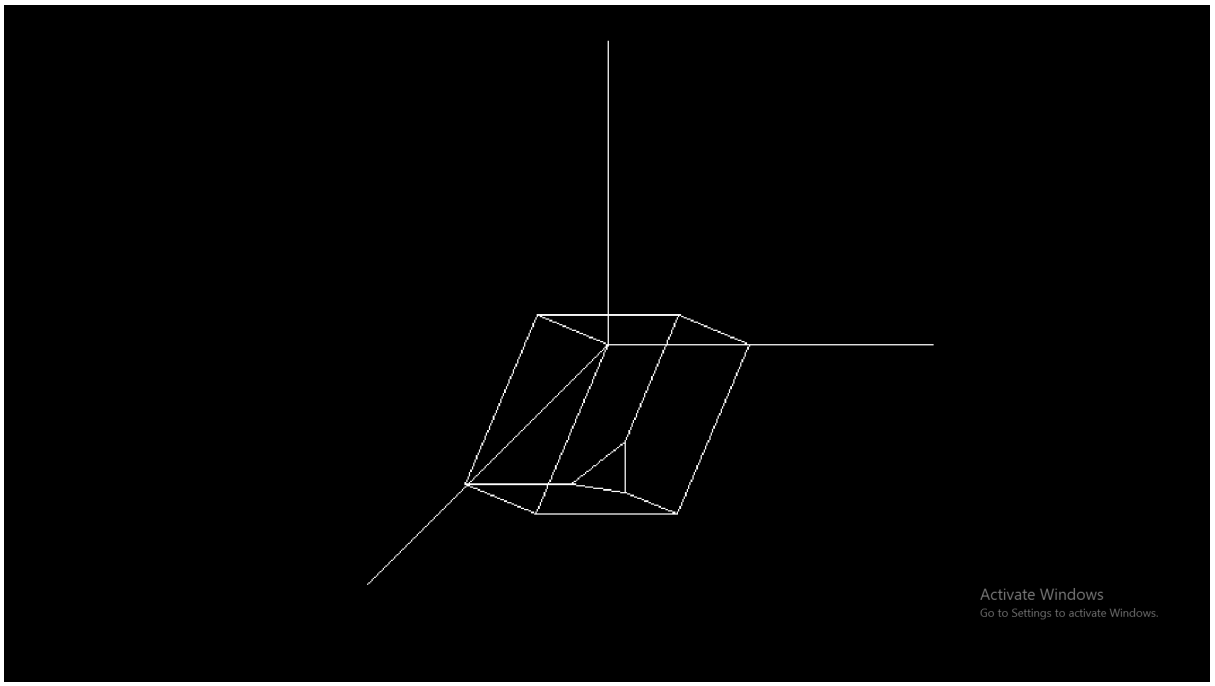




```
1:==> Draw Cube
2:==> Scaling
3:==> Rotation
4:==> Reflection
5:==> Translation
6:==> Perspective Projection
7:==> Exit
 Enter Your Choice :=3_
```



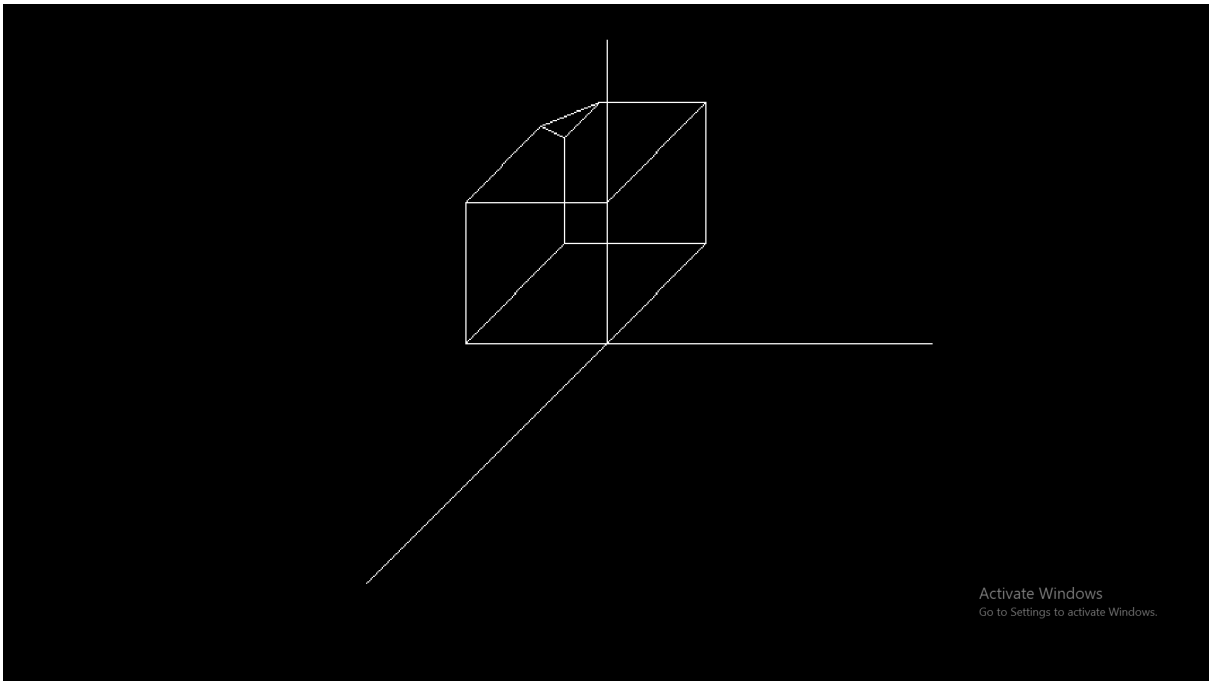
```
-=[ Rotation About ]=-
1:==> X-Axis
2:==> Y-Axis
3:==> Z-Axis
Enter Your Choice
1
 Enter The Angle 45
```




```
1:==> Draw Cube
2:==> Scaling
3:==> Rotation
4:==> Reflection
5:==> Translation
6:==> Perspective Projection
7:==> Exit
 Enter Your Choice :=4
```

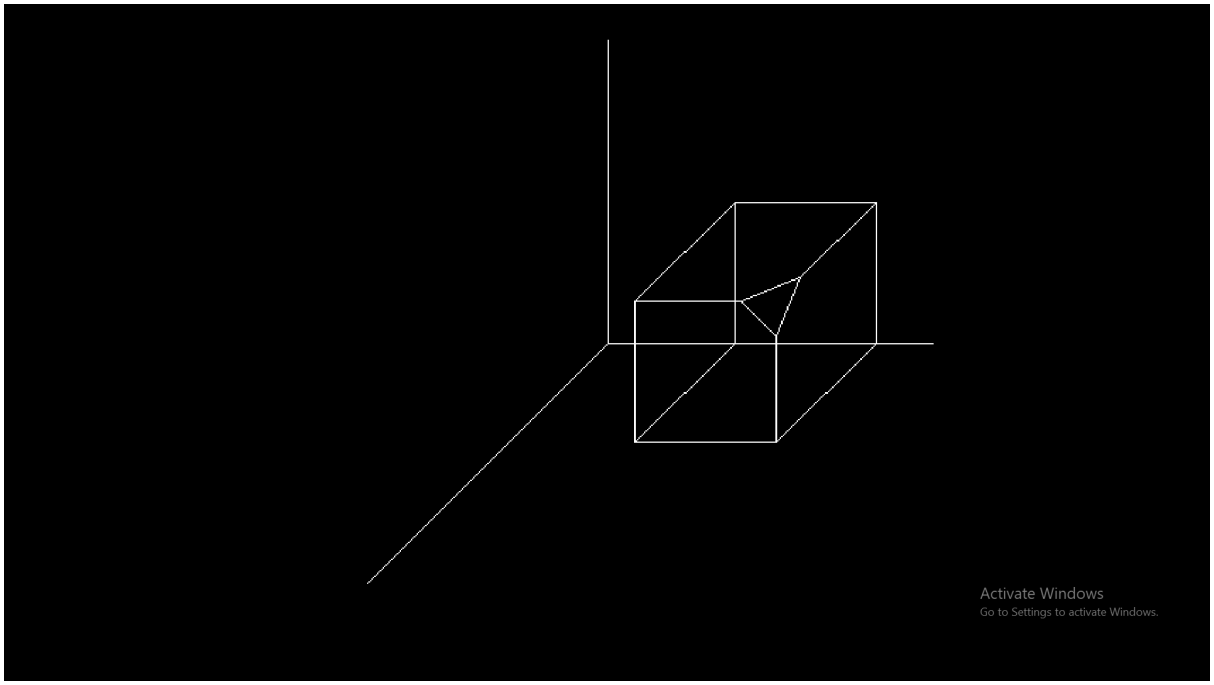
```
-=[ Reflection About ]=-
1:==> X-Axis
2:==> Y-Axis
3:==> Z-Axis
 Enter Your Choice
2_
```





```
1:==> Draw Cube
2:==> Scaling
3:==> Rotation
4:==> Reflection
5:==> Translation
6:==> Perspective Projection
7:==> Exit
 Enter Your Choice :=5
Enter The Translation Factors
20 30 40_
```

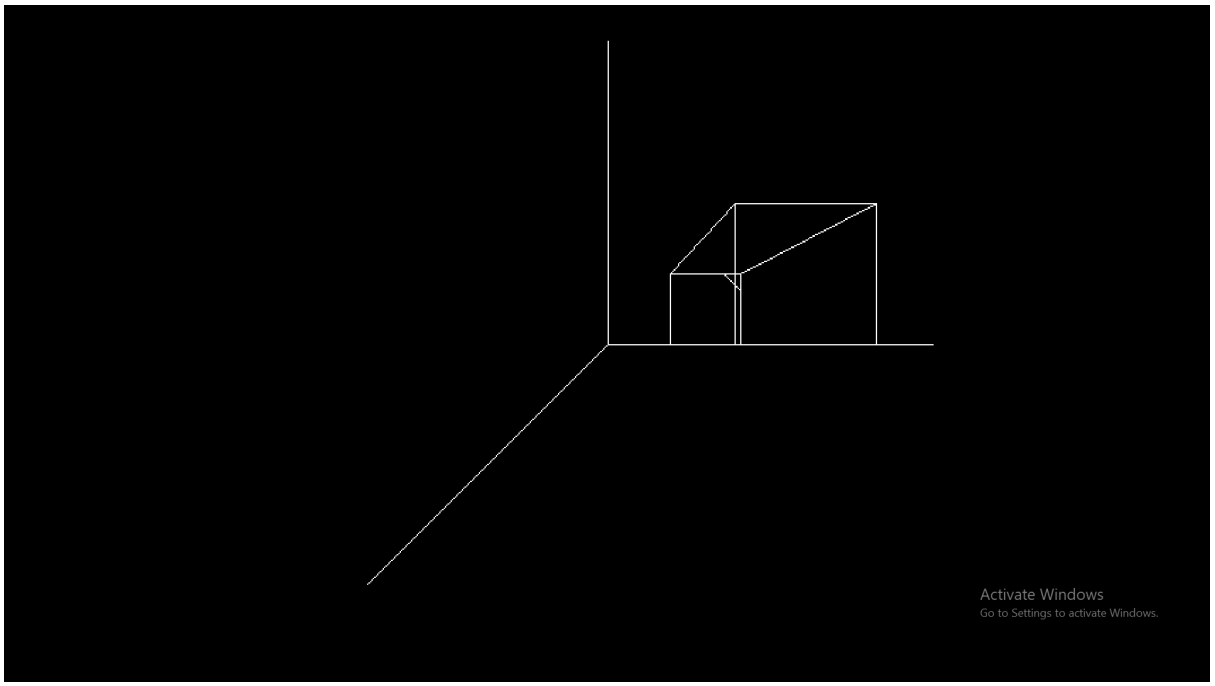




```
1:==> Draw Cube
2:==> Scaling
3:==> Rotation
4:==> Reflection
5:==> Translation
6:==> Perspective Projection
7:==> Exit
 Enter Your Choice :=6_
```



```
-=[ Perspective Projection About ]=-
1:==> X-Axis
2:==> Y-Axis
3:==> Z-Axis
 Enter Your Choice :=
3
 Enter R :=10 20 30_
```




8. Write a program to draw Hermite /Bezier curve.

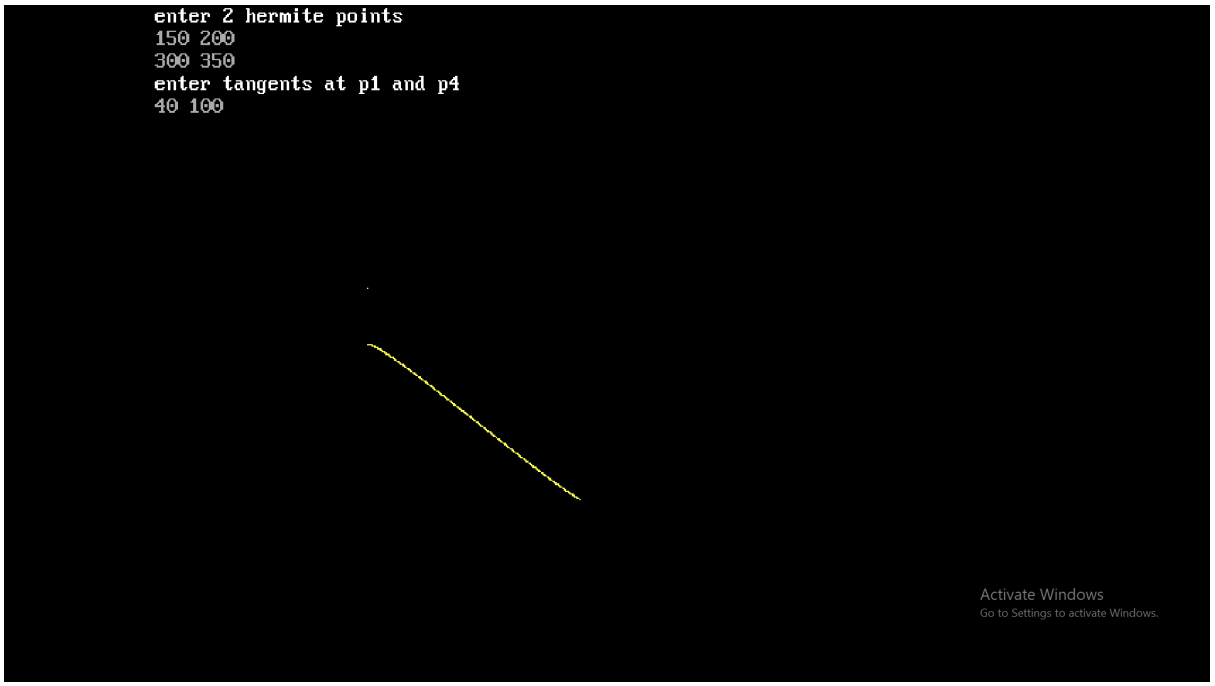
**Hermite**

```
#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>



struct point
```

```
{
 int x,y;
};

void hermite(point p1,point p4,double r1,double r4)
{
        float x,y,t;
        for(t=0.0;t<=1.0;t+=0.001)
        {

x=(2*t*t*t-3*t*t+1)*p1.x+(-2*t*t*t+3*t*t)*p4.x+(t*t*t-2*t*t+t
)*r1+(t*t*t-t*t)*r4;

y=(2*t*t*t-3*t*t+1)*p1.y+(-2*t*t*t+3*t*t)*p4.y+(t*t*t-2*t*t+1
)*r1+(t*t*t-t*t)*r4;
            putpixel(x,y,YELLOW);
        }
}
```

```
void main()
{
        /* request auto detection */
        int gdriver = DETECT, gmode, errorcode;

        /* initialize graphics and local variables */
        initgraph(&gdriver, &gmode, "C://TURBOC3//BGI");

        /* read result of initialization */
        errorcode = graphresult();

        /* an error occurred */
        if (errorcode != grOk)
        {
            printf("Graphics error: %s\n",
grapherrormsg(errorcode));
            printf("Press any key to halt:");
            getch();
            exit(1);
```

```
        }



        double r1,r4;

        point p1,p2;

        cout<<"enter 2 hermite points"<<endl;

        cin>>p1.x>>p1.y>>p2.x>>p2.y;

        cout<<"enter tangents at p1 and p4"<<endl;

        cin>>r1>>r4;

        hermite(p1,p2,r1,r4);

        putpixel(p1.x,p1.y,WHITE);

        putpixel(p2.x,p2.y,WHITE);

        getch();

        closegraph();

}
```

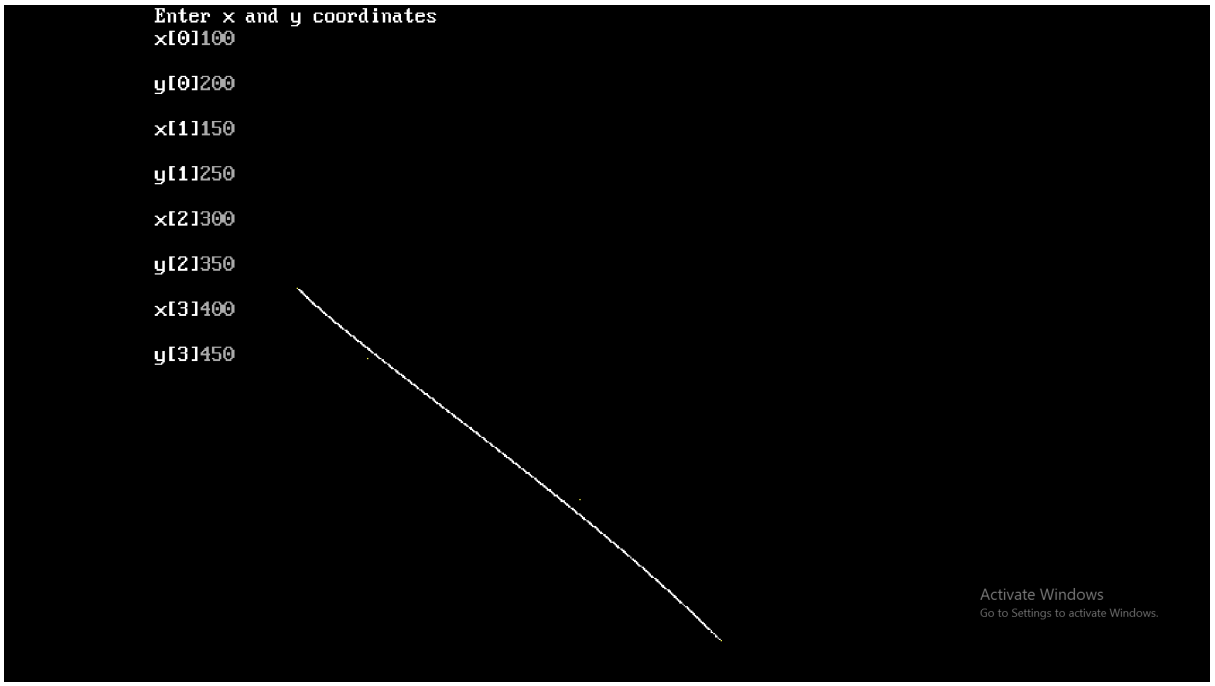



## Bezier

```
#include<stdio.h>

#include<graphics.h>

#include<iostream.h>

#include<conio.h>

#include<stdlib.h>

#include<math.h>


void bezier(int x[4], int y[4])

{

        double t;

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

        {

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

```
            double
yt=pow(1-t,3)*y[0]+3*t*pow(1-t,2)*y[1]+3*pow(t,2)*(1-t)*y[2]+p
ow(t,3)*y[3];
            putpixel(xt,yt,WHITE);
        }

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

void main()
{
        /* request auto detection */
        int gdriver = DETECT, gmode, errorcode;

        /* initialize graphics and local variables */
        initgraph(&gdriver, &gmode, "C://TURBOC3//BGI");

        /* read result of initialization */
        errorcode = graphresult();

        /* an error occurred */
        if (errorcode != grOk)
```

```
        {
            printf("Graphics error: %s\n",
grapherrormsg(errorcode));
            printf("Press any key to halt:");
            getch();
            exit(1);
        }

        int x[4],y[4];
        int i;
        cout<<"Enter x and y coordinates"<<endl;

        for(i=0;i < 4;i++)
        {
            cin>>x[i];
            cout<<endl;
            cin>>y[i];
        }
        bezier(x,y);
}
```


Enter x and y coordinates
x[0]100
y[0]200
x[1]150
y[1]250
x[2]300
y[2]350
x[3]400
y[3]450
Activate Windows
Go to Settings to activate Windows.