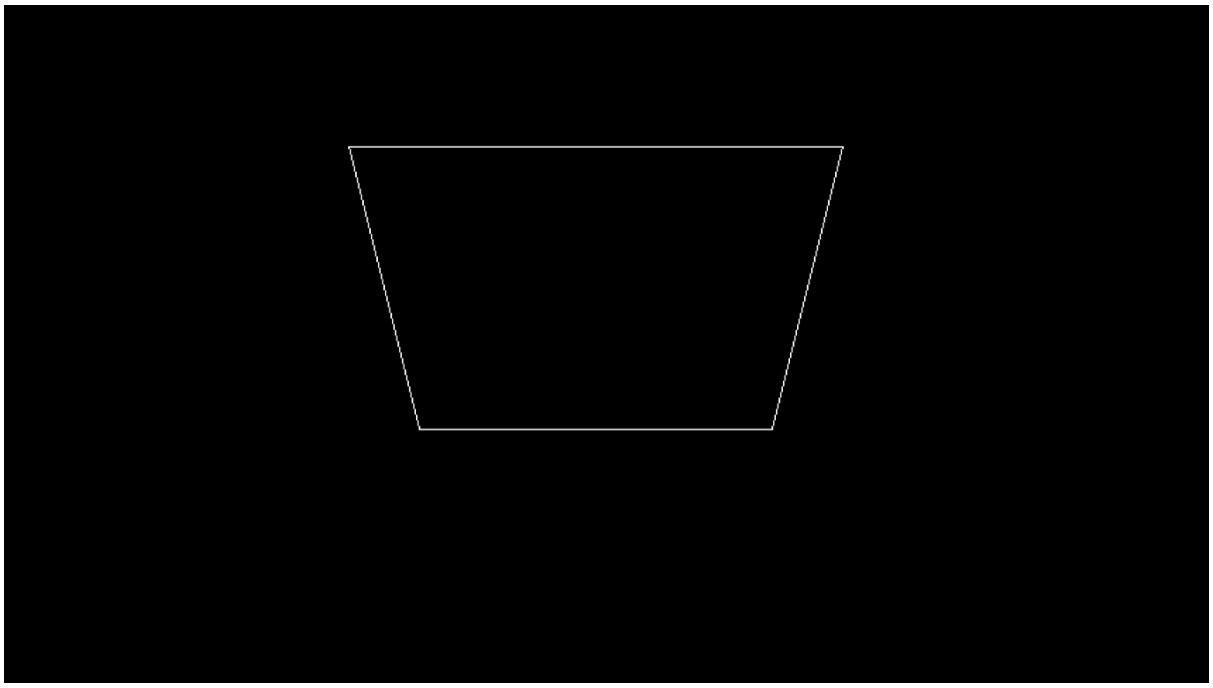
Practical No. 1.1

Aim:- Write a graphics program for moving boat.

Source code:

```
#include<iostream.h>
#include<conio.h>;
#include<graphics.h>
#include<math.h>
#include<dos.h>
#include<stdlib.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
for(int i=0;i<=300;i++)
{
cleardevice();
line(100+i,100,150+i,300);
line(150+i,300,400+i,300);
line(400+i,300,450+i,100);
line(100+i,100,450+i,100);
}
getch();
}
```

Output:



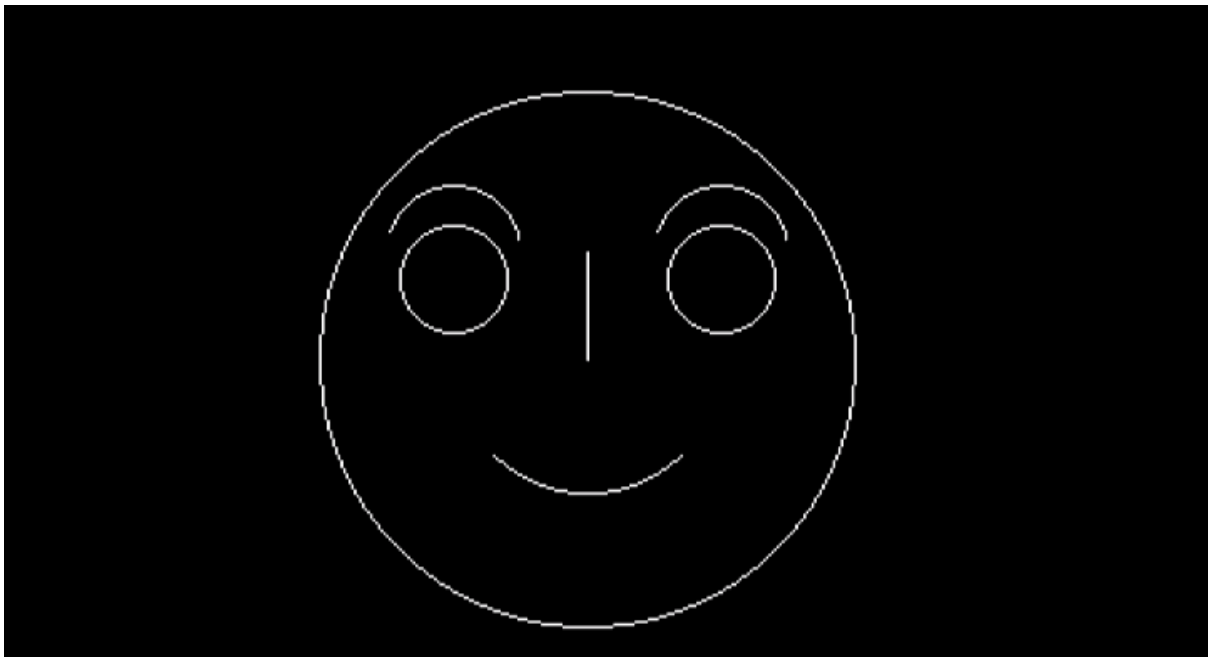
Practical No. 1.2

Aim:- Write a graphics program to draw smiley face.

Source code:

```
#include<iostream.h>
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
void main()
{
 clrscr();
 int gd,gm;
 detectgraph(&gd,&gm);
 initgraph(&gd,&gm,"C:\\Turboc3\\bgi");
 int x,y;
 x=getmaxx()/2;
 y=getmaxy()/2;
 circle(x,y,100); //main circle
 circle(x-50,y-30,20); //left eye
 circle(x+50,y-30,20); //right eye
 arc(x-50,y-40,10,160,25); //left eye arc
 arc(x+50,y-40,10,160,25); //right eye arc
 arc(x,y,225,315,50);//smile arc
 line(x,y,x,y-40); //nose
getch();
}
```

Output:



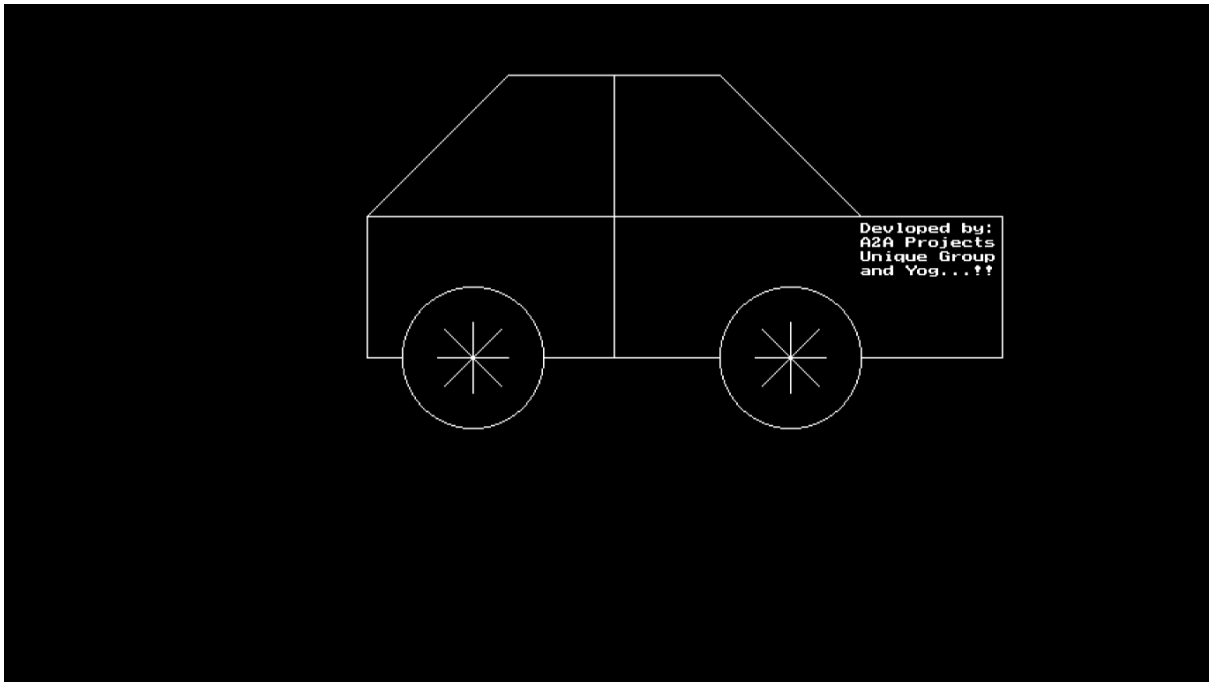
Practical No. 1.3

Aim:- Write a graphics program for car.

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<dos.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm, "c:\\TurboC3\\BGI");
for(int i=0;i<=100;i++)
{
cleardevice();
line(150+i,50,300+i,50);  //line1
line(50+i,150,150+i,50); //line2
line(50+i,150,400+i,150);
line(225+i,50,225+i,250);
line(400+i,150,300+i,50);  //line3
line(400+i,150,500+i,150);  //line4
line(500+i,150,500+i,250);   //line5
line(500+i,250,400+i,250);   //line6
circle(350+i,250,50);  //line7
line(350+i,225,350+i,275);
line(325+i,250,375+i,250);
line(330+i,230,370+i,270);
line(330+i,270,370+i,230);
line(300+i,250,175+i,250);   //line8
circle(125+i,250,50);  //line9
line(100+i,250,150+i,250);
line(125+i,225,125+i,275);
line(105+i,230,145+i,270);
line(105+i,270,145+i,230);
line(50+i,250,75+i,250);   //line10
line(50+i,150,50+i,250);   //line11
outtextxy(400+i,155,"Devloped by:");
outtextxy(400+i,165,"A2A Projects");
outtextxy(400+i,175, "Unique Group");
outtextxy(400+i,185, "and Yog...!!");
delay(100);
}
getch();
}
```

Output:



Practical No. 2.1

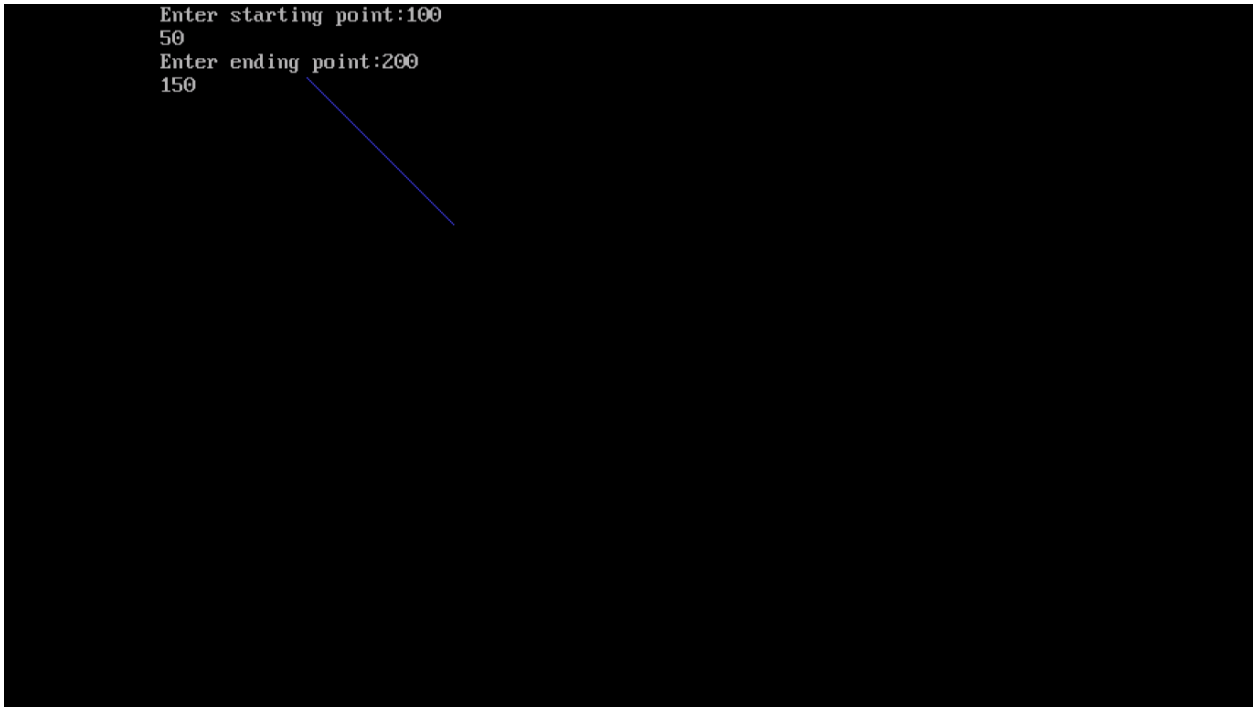
Aim:- Implementation of DDA Line.

Source code:

```
#include<iostream.h>
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
#include<dos.h>
#include<stdlib.h>
void main()
{
 clrscr();
 int gd,gm;
 detectgraph(&gd,&gm);
 initgraph(&gd,&gm,"C:\\Turboc3\\bgi");
 int x1,y1,x2,y2,dx,dy,length,x,y,i;
 cout<<"Enter starting point:";
 cin>>x1>>y1;
 cout<<"Enter ending point:";
 cin>>x2>>y2;
 dx=abs(x2-x1);
 dy=abs(y2-y1);
 if(dx>=dy)
 {
   length=dx;
 }
 else
 {
   length=dy;
 }
 dx=(x2-x1)/length;
 dy=(y2-y1)/length;
 x=x1+0.5;
 y=y1+0.5;
 i+1;
 while(i<=length)
 {
   putpixel(x,y,25);
   x=x+dx;
   y=y+dy;
   i=i+1;
   delay(100);
 }

 getch();
}
```

Output:



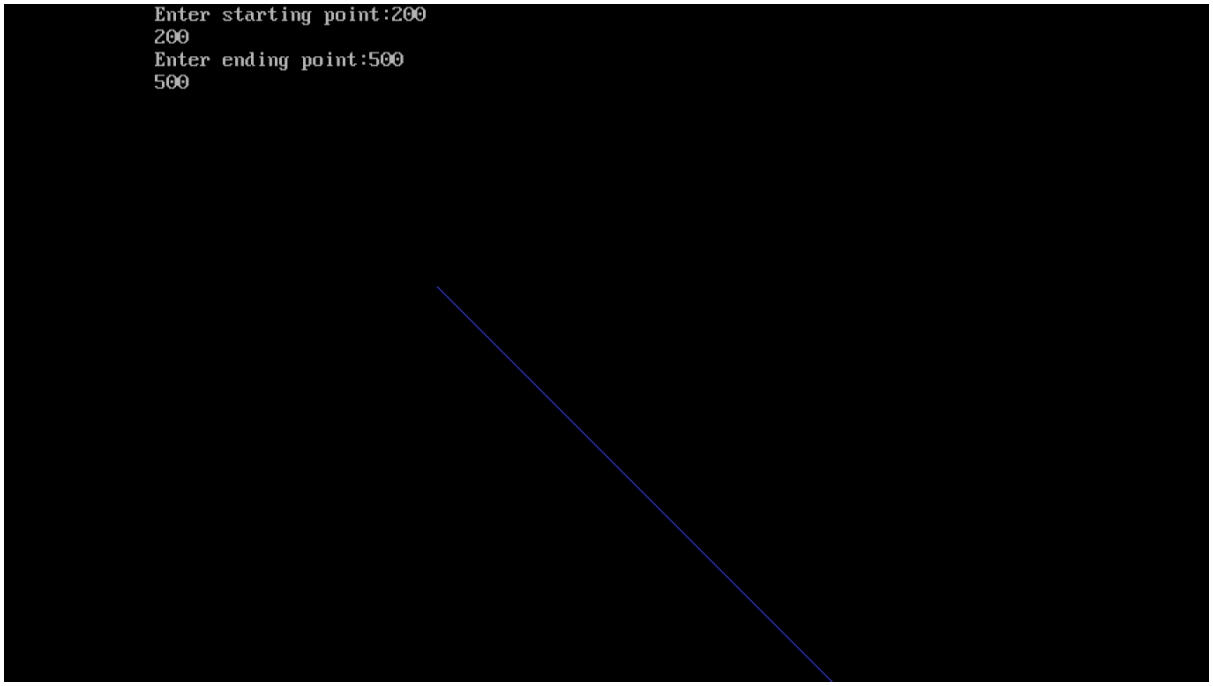
Practical No. 2.2

Aim: Implementation of Bresenham's line.

Source code:

```
#include<iostream.h>
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
#include<dos.h>
#include<stdlib.h>
void main()
{
  int gd,gm;
  int x1,y1,x2,y2,dx,dy,x,y,e,i;
  clrscr();
  detectgraph(&gd,&gm);
  initgraph(&gd,&gm,"c:\\turboc3\\bgi");
  cout<<"Enter starting point:";
  cin>>x1>>y1;
  cout<<"Enter ending point:";
  cin>>x2>>y2;
  dx=abs(x2-x1);
  dy=abs(y2-y1);
  x=x1;
  y=y1;
  e=2*dy-dx;
  i=1;
  do
  {
  putpixel(x,y,25);
  while(e>=0)
  {
  y=y+1;
  e=e-2*dx;
  }
  x=x+1;
  e=e+2*dy;
  i=i+1;
  delay(100);
  }while(i<=dx);
  getch();
}
```

Output:



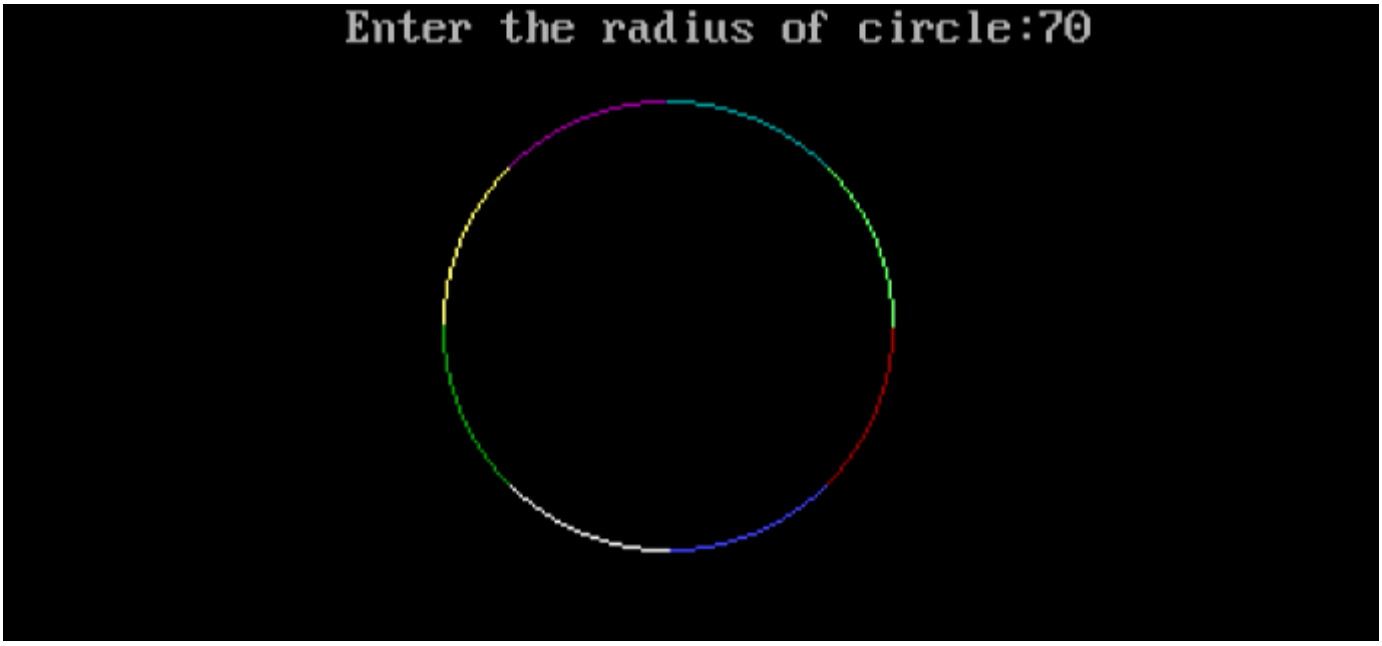
Practical No.3.1

Aim: Write a graphics program to implement mid-point circle algorithm.

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<dos.h>
#include<math.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
int r,x,y;
double d;
cout<<"Enter the radius of circle:";
cin>>r;
x=0;y=r;
d=1.25-r;
do
{
putpixel(100+x,100+y,25);//1
putpixel(100+y,100+x,20);//2
putpixel(100-x,100+y,15);//3
putpixel(100+y,100-x,10);//4
putpixel(100-x,100-y,5);//5
putpixel(100-y,100-x,30);//6
putpixel(100+x,100-y,35);//7
putpixel(100-y,100+x,50);//8
if(d<0)
{
x=x+1;
y=y;
d=d+2*x+1;
}
else
{
x=x+1;
y=y-1;
d=d+2*x-2*y+1;
}
delay(100);
}while(x<y);
getch();
}
```

Output:



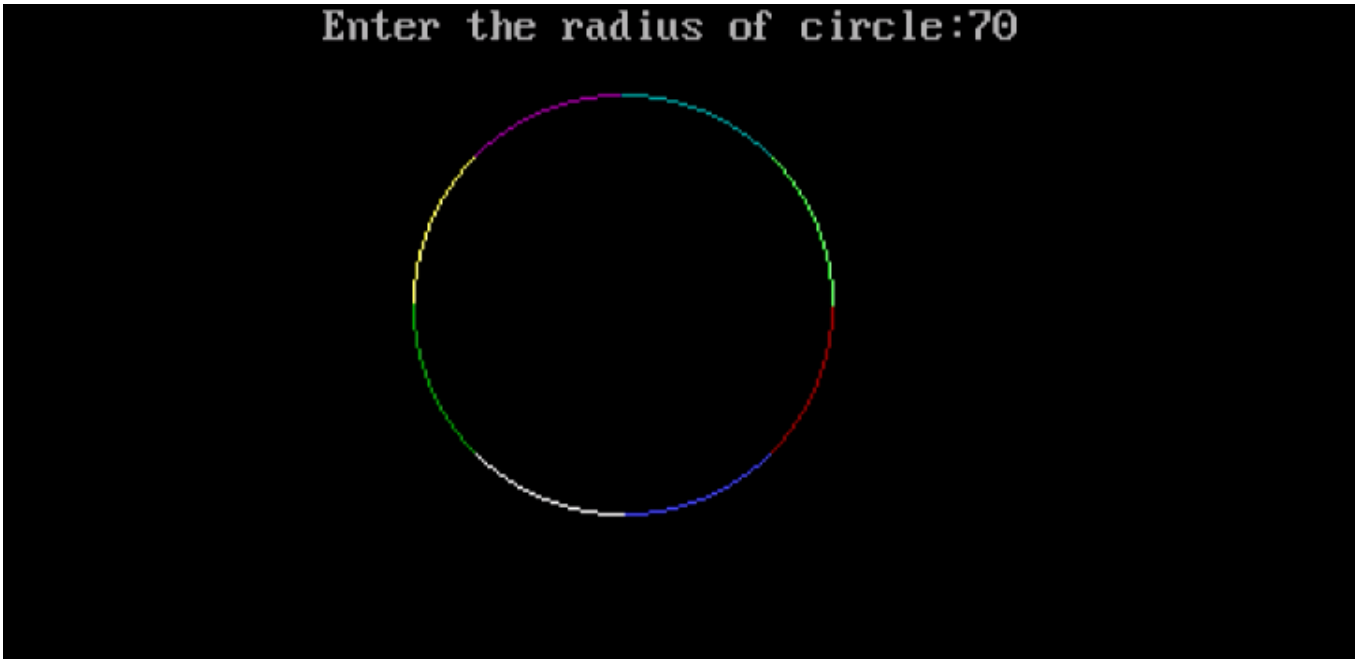
Practical No.3.2

Aim:- Write a graphics program to implement bresenham circle algorithm.

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<dos.h>
#include<math.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
int r,x,y,d;
cout<<"Enter the radius of circle:";
cin>>r;
x=0;y=r;
d=3-2*r;
do
{
putpixel(100+x,100+y,25);//1
putpixel(100+y,100+x,20);//2
putpixel(100-x,100+y,15);//3
putpixel(100+y,100-x,10);//4
putpixel(100-x,100-y,5);//5
putpixel(100-y,100-x,30);//6
putpixel(100+x,100-y,35);//7
putpixel(100-y,100+x,30);//8
if(d<0)
{
d=d+4*x+6;
}
else
{
d=d+4*(x-y)+10;
y=y-1;
}
x=x+1;
delay(100);
}while(x<y);
getch();
}
```

Output:



Practical No. 3.3

Aim: Write a graphics program for implementation of Arc.

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm, "c:\\TurboC3\\BGI");
//arc(250,150,0,300,40);
int x,y,stangle,endangle,r;
x=getmaxx()/2;
y=getmaxy()/2;
cout<<"Enter starting angle"<<endl;
cin>>stangle;
cout<<"Enter ending angle:"<<endl;
cin>>endangle;
cout<<"Enter the radius of arc:"<<endl;
cin>>r;
arc(x,y,stangle,endangle,r);
getch();
}
```

Output:

Practical No. 4.1

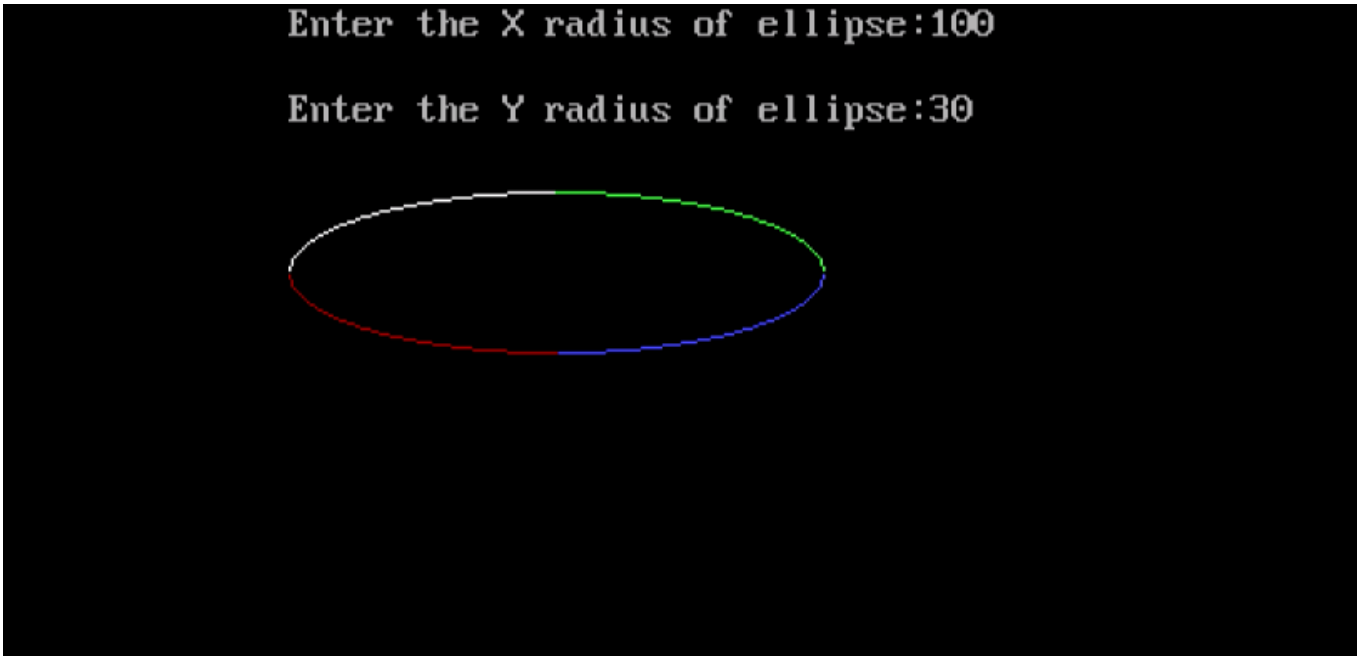
Aim: Write a graphics program to implement ellipse drawing algorithm.

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<dos.h>
#include<math.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
int x,y;
long d1,d2,dx,dy,rx,ry;
cout<<"Enter the X radius of ellipse:";
cin>>rx;
cout<<"\nEnter the Y radius of ellipse:";
cin>>ry;
x=0;y=ry;
d1=(ry*ry)-(rx*rx)*ry+(0.25*rx*rx);
dx=2*ry*ry*x;
dy=2*rx*rx*y;
do
{
putpixel(100+x,100+y,25);//1
putpixel(100-x,100+y,20);//2
putpixel(100-x,100-y,15);//3
putpixel(100+x,100-y,10);//4
if(d1<0)
{
x=x+1;
y=y;
dx=dx+(2*ry*ry);
d1=d1+dx+(ry*ry);
}
else
{
x=x+1;
y=y-1;
dx=dx+(2*ry*ry);
dy=dy-(2*rx*rx);
d1=d1+dx-dy+(ry*ry);
}
delay(100);
}while(dx<dy);
d2=(ry*ry)*(x+0.5)*(x+0.5)+(rx*rx)*(y-1)*(y-1)-(rx*rx)*(ry*ry);
do
{
putpixel(100+x,100+y,25);//1
putpixel(100-x,100+y,20);//2
```

```
putpixel(100-x,100-y,15);//3
putpixel(100+x,100-y,10);//4
if(d2>0)
{
x=x;
y=y-1;
dy=dy-(2*rx*rx);
d2=d2-dy+(rx*rx);
}
else
{
x=x+1;
y=y-1;
dy=dy-(2*rx*rx);
dx=dx+(2*ry*ry);
d2=d2+dx-dy+(rx*rx);
}
delay(100);
}while(y>0);
getch();
}
```

Output:



Practical No. 5.1

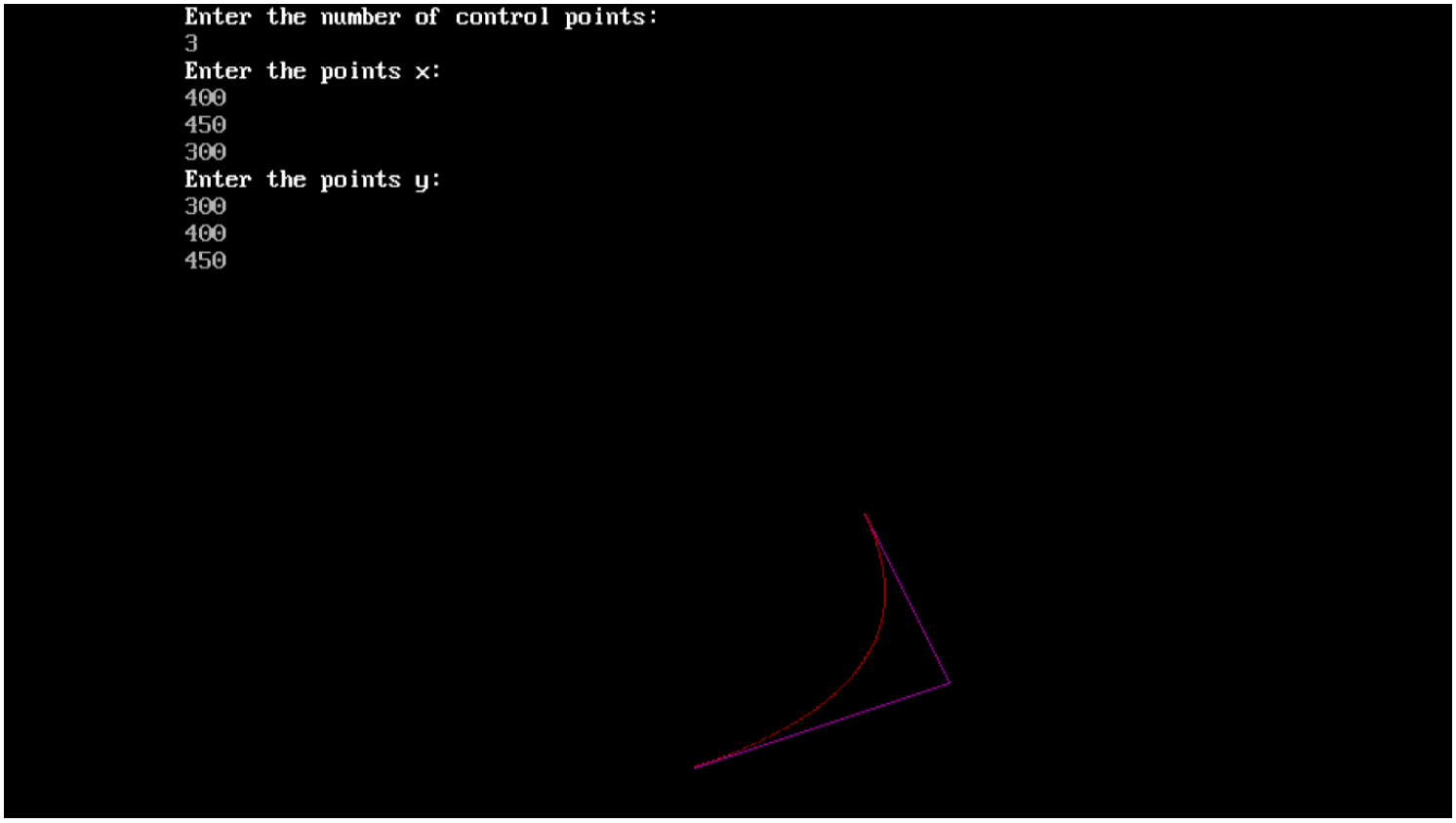
Aim: Implementation of drawing curve using Bezier Curve.

Source code:

```
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>
#define MAX 10
float mypower(float n,int k)
{
if(n==0.0)
return 1;
else
{
float power=1.0;
for(int i=1;i<=k;i++)
power*=n;
return power;
}
}
int fact(int n)
{
int prod = 1;
for(int i=n;i>1;i--)
prod*=i;
return prod;
}
int c(int n,int k)
{
return fact(n)/(fact(k)*fact(n-k));
}
float bez(int n,int k,float u)
{
return (c(n,k)*mypower(u,k)*mypower(1-u,n-k));
}
void main()
{
clrscr();
int px[MAX],py[MAX],n;
int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
cout<<"Enter the number of control points:\n";
cin>>n;
cout<<"Enter the points x:\n";
for(int i=0;i<n;i++)
cin>>px[i];
cout<<"Enter the points y:\n";
for(int j=0;j<n;j++)
cin>>py[j] ;
setcolor(5);
for(i=0;i<n-1;i++)
line(px[i],py[i],px[i+1],py[i+1]);
```

```
setcolor(15);
float u = 0.0;
while(u<=1)
{
float sumx=0,sumy=0;
for(int k=0;k<n;k++)
{
sumx+=px[k]*bez(n-1,k,u);
sumy+=py[k]*bez(n-1,k,u);
}
setcolor(15);
putpixel(sumx,sumy,RED);
u=u+0.005;
}
getch();
}
```

Output:



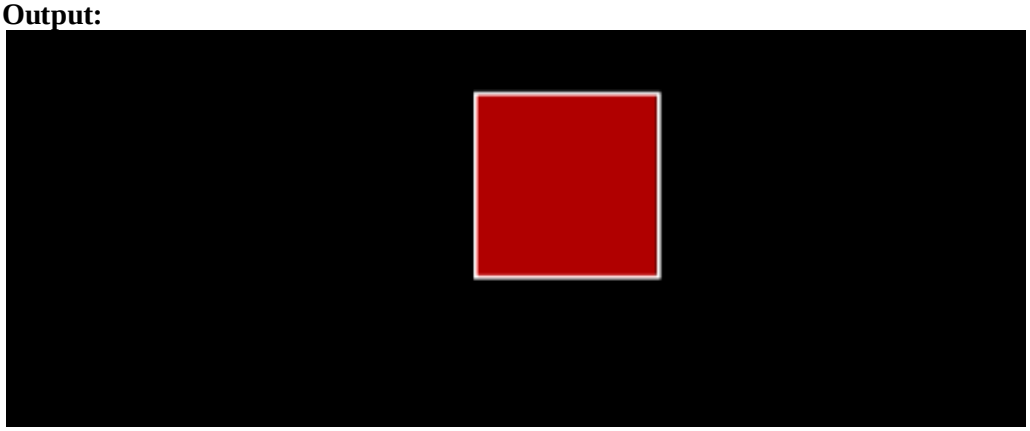
Practical No.: 6.1

Aim: Write a graphics program to implement of polygon filling algorithm using boundary fill(4 connectivity).

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
#include<dos.h>
#include<graphics.h>
void boundary_fill(int x,int y,int f_color,int b_color)
{
if(getpixel(x,y)!=f_color && getpixel(x,y)!=b_color)
{
putpixel(x,y,f_color);
boundary_fill(x+1,y,f_color,b_color);
boundary_fill(x-1,y,f_color,b_color);
boundary_fill(x,y-1,f_color,b_color);
boundary_fill(x,y+1,f_color,b_color);
}
}
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
rectangle(50,50,100,100);
boundary_fill(55,55,4,15);
getch();
}
```

Output:



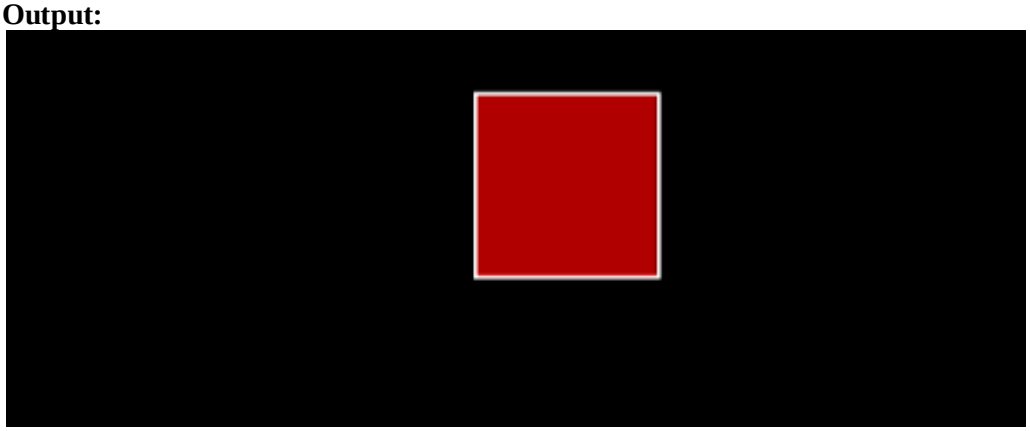
Practical No.: 6.1

Aim: Write a graphics program to implement of polygon filling algorithm using boundary fill(8 connectivity).

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
#include<dos.h>
#include<graphics.h>
void boundary_fill(int x,int y,int f_color,int b_color)
{
if(getpixel(x,y)!=f_color && getpixel(x,y)!=b_color)
{
putpixel(x,y,f_color);
boundary_fill(x+1,y,f_color,b_color);
boundary_fill(x-1,y,f_color,b_color);
boundary_fill(x,y-1,f_color,b_color);
boundary_fill(x,y+1,f_color,b_color);
boundary_fill(x+1,y-1,f_color,b_color);
boundary_fill(x-1,y-1,f_color,b_color);
boundary_fill(x+1,y+1,f_color,b_color);
boundary_fill(x-1,y+1,f_color,b_color);
}
}
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
rectangle(50,50,100,100);
boundary_fill(55,55,4,15);
getch();
}
```

Output:



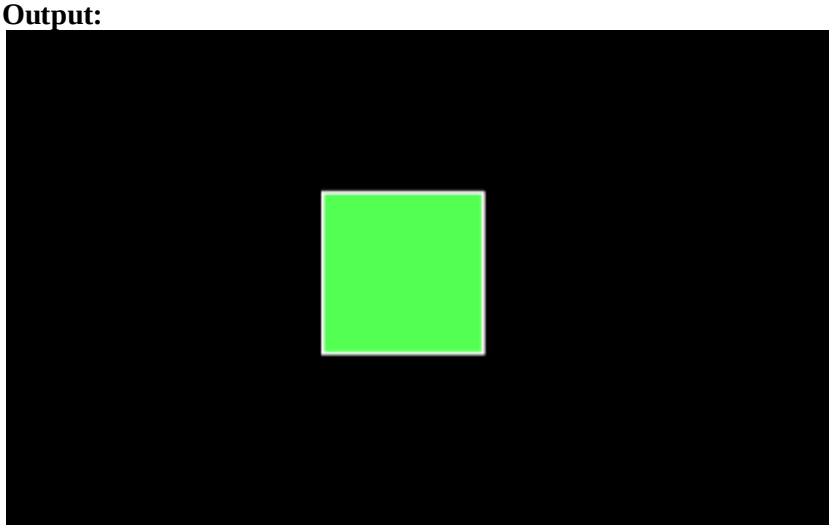
Practical No. 6.2

Aim: Write a graphics program to impalement filling polygon using flood fill algorithm(4 connectivity).

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
#include<dos.h>
#include<graphics.h>
void flood_fill(int x,int y,int o_color,int n_color)
{
if(getpixel(x,y)==o_color)
{
putpixel(x,y,n_color);
flood_fill(x+1,y,o_color,n_color);
flood_fill(x-1,y,o_color,n_color);
flood_fill(x,y-1,o_color,n_color);
flood_fill(x,y+1,o_color,n_color);
}
}
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
rectangle(50,50,100,100);
flood_fill(55,55,0,10);
getch();
}
```

Output:



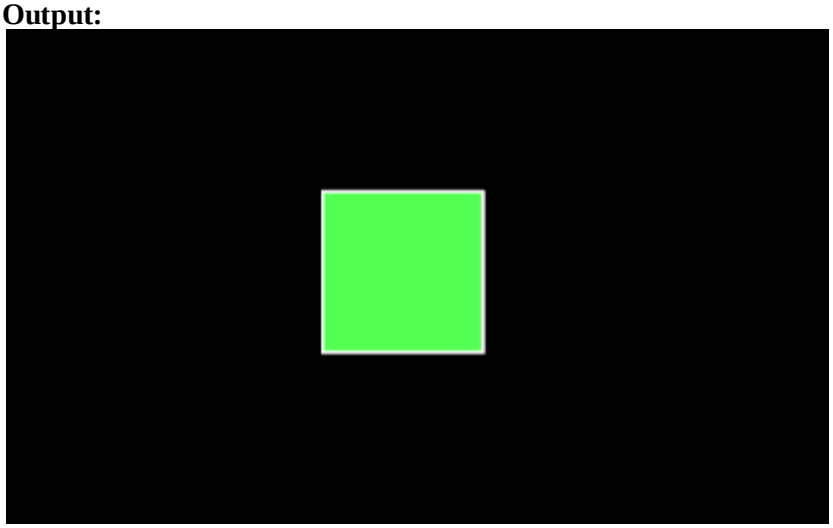
Practical No. 6.2

Aim: Write a graphics program to impalement filling polygon using flood fill algorithm(8 connectivity).

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
#include<dos.h>
#include<graphics.h>
void flood_fill(int x,int y,int o_color,int n_color)
{
if(getpixel(x,y)==o_color)
{
putpixel(x,y,n_color);
flood_fill(x+1,y,o_color,n_color);
flood_fill(x-1,y,o_color,n_color);
flood_fill(x,y-1,o_color,n_color);
flood_fill(x,y+1,o_color,n_color);
flood_fill(x+1,y-1,o_color,n_color);
flood_fill(x-1,y-1,o_color,n_color);
flood_fill(x+1,y+1,o_color,n_color);
flood_fill(x-1,y+1,o_color,n_color);
}
}
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
rectangle(50,50,100,100);
flood_fill(55,55,0,10);
getch();
}
```

Output:



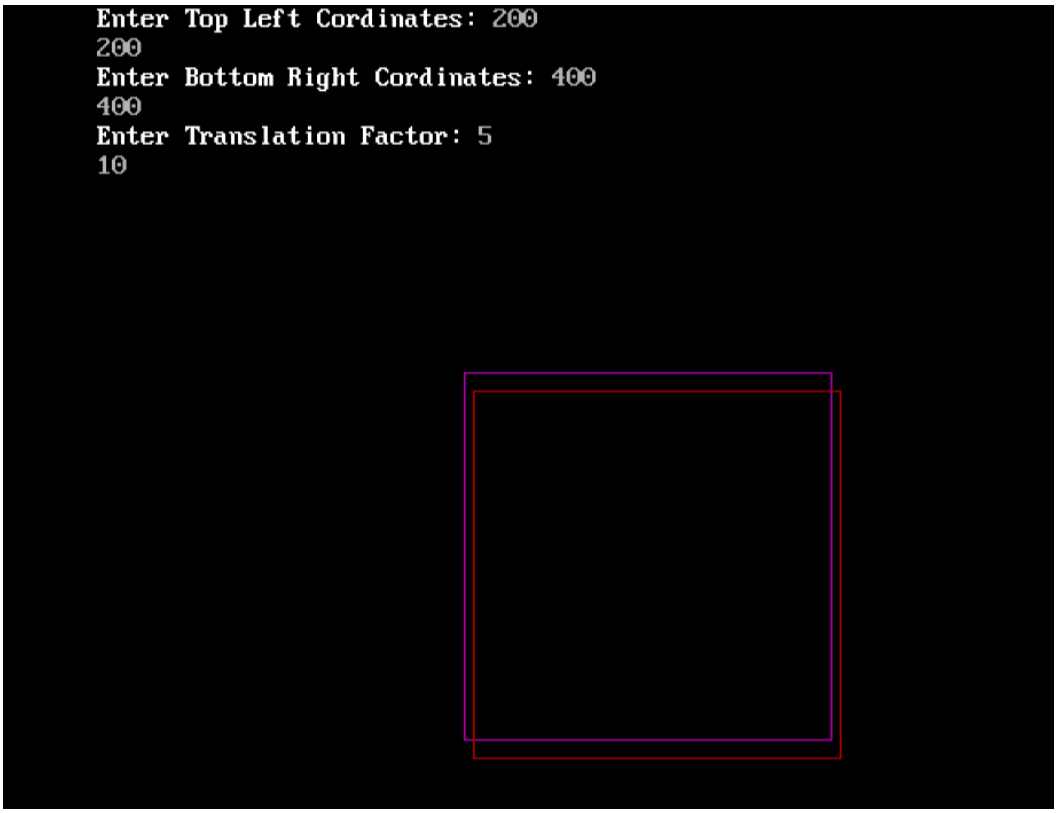
Practical No.7.1

Aim:- Write a graphics program for implement 2-D translation of rectangle.

Source Code:-

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<stdlib.h>
#include<dos.h>
void main()
{
        int gd,gm,xmin,ymin,xmax,ymax;
        float sx,sy;
        clrscr();
        detectgraph(&gd,&gm);
        initgraph(&gd,&gm,"c:\\turboc3\\bgi");
        cout<<"Enter Top Left Cordinates";
        cin>>xmin>>ymin;
        cout<<"Enter Bottom Right Cordinates";
        cin>>xmax>>ymax;
        setcolor(5);
        rectangle(xmin,ymin,xmax,ymax);
        cout<<"Enter Translation Factor";
        cin>>tx>>ty;
        int x1=xmin+tx;
        int y1=ymin+ty;
        int x2=xmax+tx;
        int y2=ymax+ty;
        setcolor(4);
        rectangle(x1,y1,x2,y2);
        getch();
}
```

Output:-



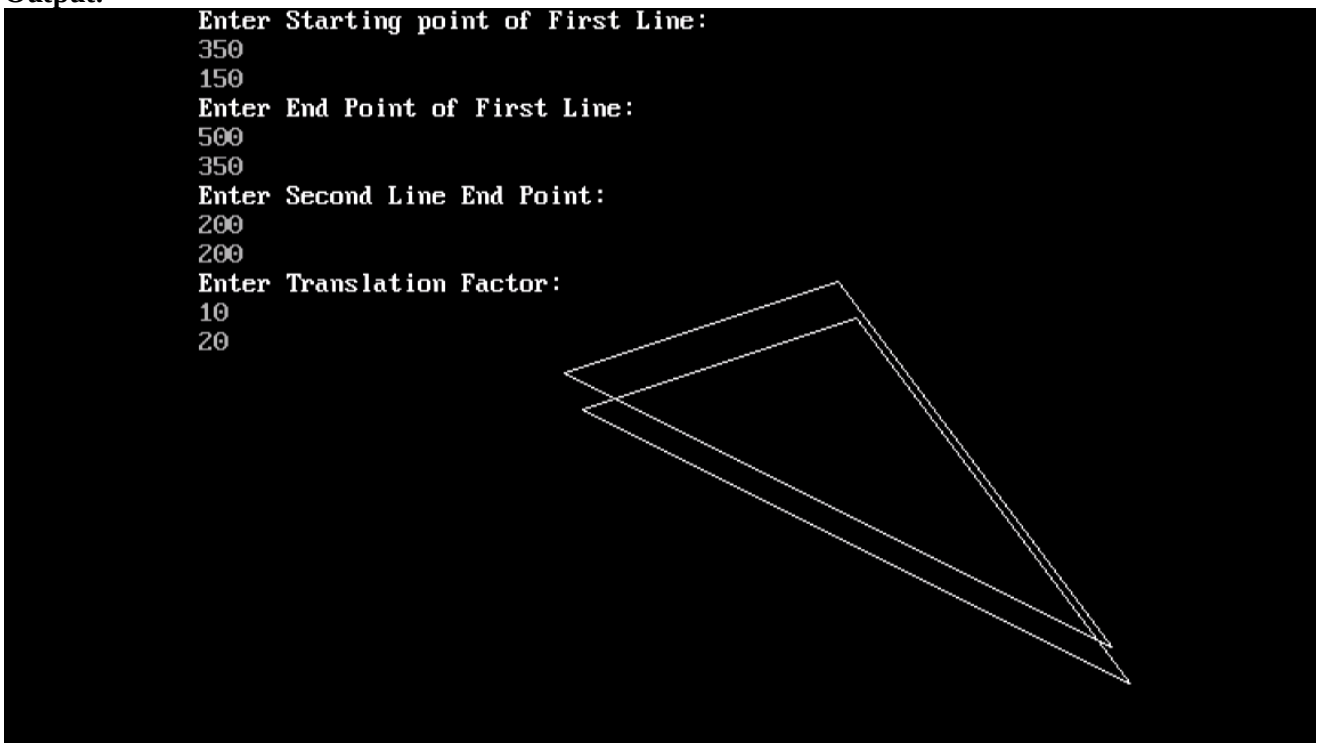
Practical No. 7.1

Aim:- Write a graphics program for translate a triangle.

Source code:-

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<stdlib.h>
#include<dos.h>
void main()
{
	int gd,gm,x,y,x2,y2,x3,y3,tx,ty;
	clrscr();
	detectgraph(&gd,&gm);
	initgraph(&gd,&gm,"c:\\turboc3\\bgi");
	cout<<"Enter Starting point of First Line"<<endl;
	cin>>x>>y;
	cout<<"Enter End Point of First Line"<<endl;
	cin>>x2>>y2;
	line(x,y,x2,y2);
	cout<<"Enter Second Line End Point"<<endl;
	cin>>x3>>y3;
	line(x,y,x3,y3);
	line(x2,y2,x3,y3);
	cout<<"Enter Translation Factor"<<endl;
	cin>>tx>>ty;
	x=x+tx;
	x2=x2+tx;
	x3=x3+tx;
	y=y+ty;
	y2=y2+ty;
	y3=y3+ty;
	line(x,y,x2,y2);
	line(x2,y2,x3,y3);
	line(x,y,x3,y3);
	getch();
}
```

Output:


Enter Starting point of First Line:
350
150
Enter End Point of First Line:
500
350
Enter Second Line End Point:
200
200
Enter Translation Factor:
10
20

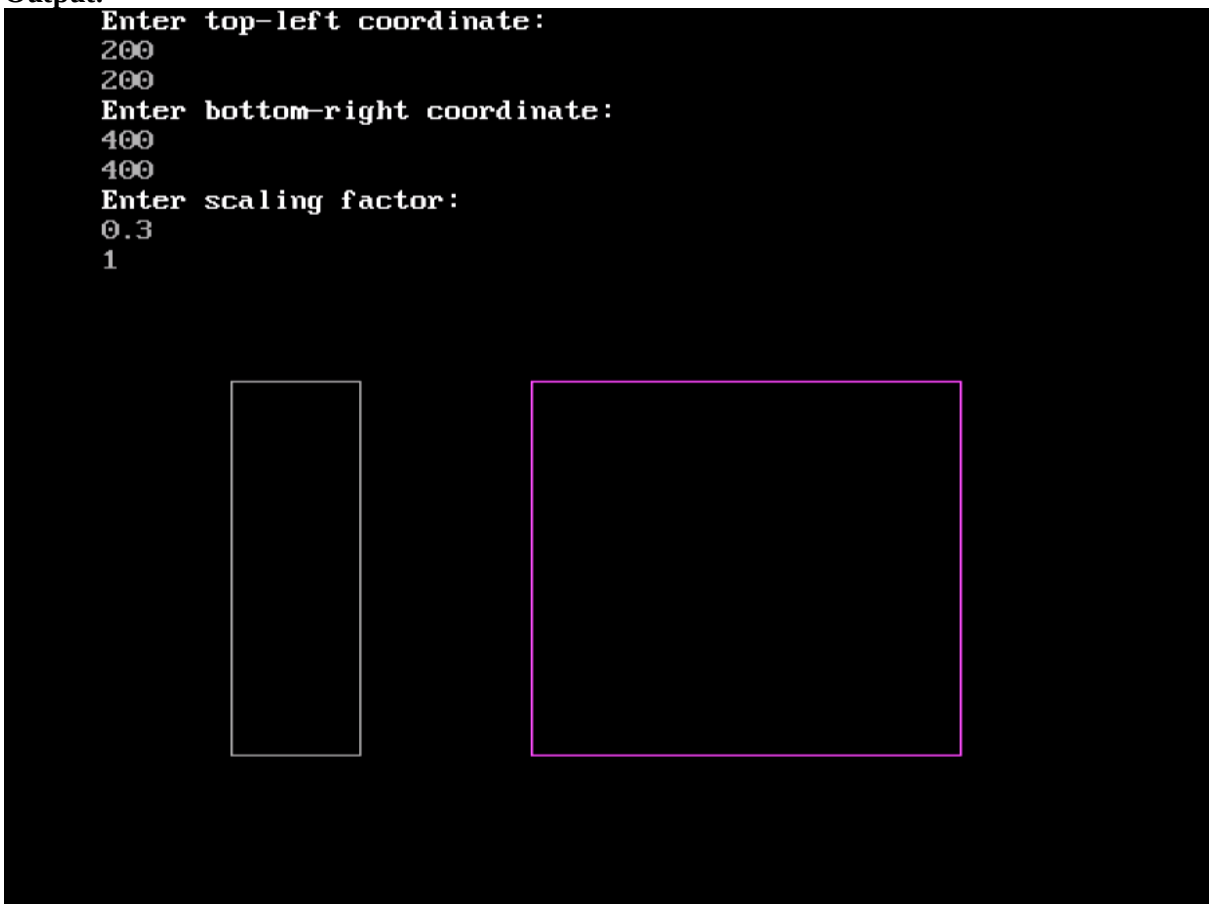
Practical No. 7.1

Aim:- Write a graphics program for scaling a rectangle.

Source Code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<dos.h>
#include<math.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
int xmin,ymin,xmax,ymax;
cout<<"Enter top-left coordinate"<<endl;
cin>>xmin>>ymin;
cout<<"Enter bottom-right coordinate"<<endl;
cin>>xmax>>ymax;
setcolor(13);
rectangle(xmin,ymin,xmax,ymax);
float sx,sy;
cout<<"Enter scaling factor"<<endl;
cin>>sx>>sy;
float x1=xmin*sx;
float y1=ymin*sy;
float x2=xmax*sx;
float y2=ymax*sy;
setcolor(23);
rectangle(x1,y1,x2,y2);
getch();
}
```

Output:


Enter top-left coordinate:
200
200
Enter bottom-right coordinate:
400
400
Enter scaling factor:
0.3
1

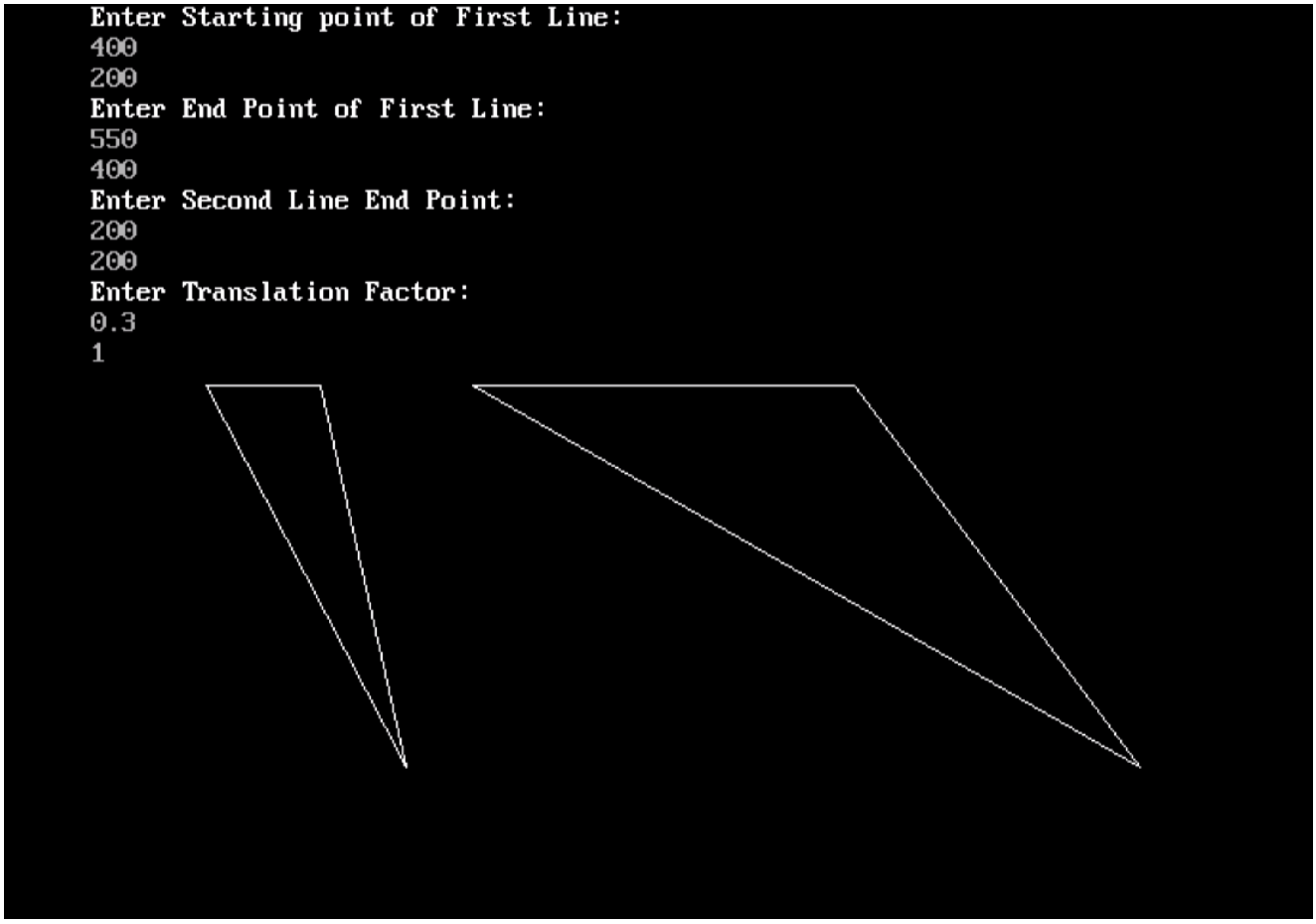
Practical No. 7.1

Aim: Write a graphics program for scaling a triangle.

Source Code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<dos.h>
#include<math.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
int xmin,ymin,xmax,ymax;
cout<<"Enter top-left coordinate: "<<endl;
cin>>xmin>>ymin;
cout<<"Enter bottom-right coordinate: "<<endl;
cin>>xmax>>ymax;
setcolor(13);
rectangle(xmin,ymin,xmax,ymax);
float sx,sy;
cout<<"Enter scaling factor: "<<endl;
cin>>sx>>sy;
float x1=xmin*sx;
float y1=ymin*sy;
float x2=xmax*sx;
float y2=ymax*sy;
setcolor(23);
rectangle(x1,y1,x2,y2);
getch();
}
```

Output:


Enter Starting point of First Line:
400
200
Enter End Point of First Line:
550
400
Enter Second Line End Point:
200
200
Enter Translation Factor:
0.3
1

Practical No. 7.1

Aim: Write a graphics program for rotating a rectangle.

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<stdlib.h>
#include<dos.h>
void main()
{
	int gd,gm;
	float xmin,ymin,xmax,ymax;
	float angle,rad;
	clrscr();
	detectgraph(&gd,&gm);
	initgraph(&gd,&gm,"c:\\turboc3\\bgi");
	cout<<"Enter Top Left Cordinates: ";
	cin>>xmin>>ymin;
	cout<<"Enter Bottom Right Cordinates: ";
	cin>>xmax>>ymax;
	setcolor(5);
	rectangle(xmin,ymin,xmax,ymax);
	cout<<"Enter Angle of Roataion: ";
	cin>>angle;
	rad=(angle*3.14)/180;
	setcolor(4);
	float x1=xmin*cos(rad)-ymin*sin(rad);
	float y1=xmin*sin(rad)+ymin*cos(rad);
	float x2=xmax*cos(rad)-ymax*sin(rad);
	float y2=xmax*sin(rad)+ymax*cos(rad);
	rectangle(x1,y1,x2,y2);
	getch();
}
```

Output:


Enter Top Left Cordinates: 100
100
Enter Bottom Right Cordinates: 300
300
Enter Angle of Roataion: 4.5

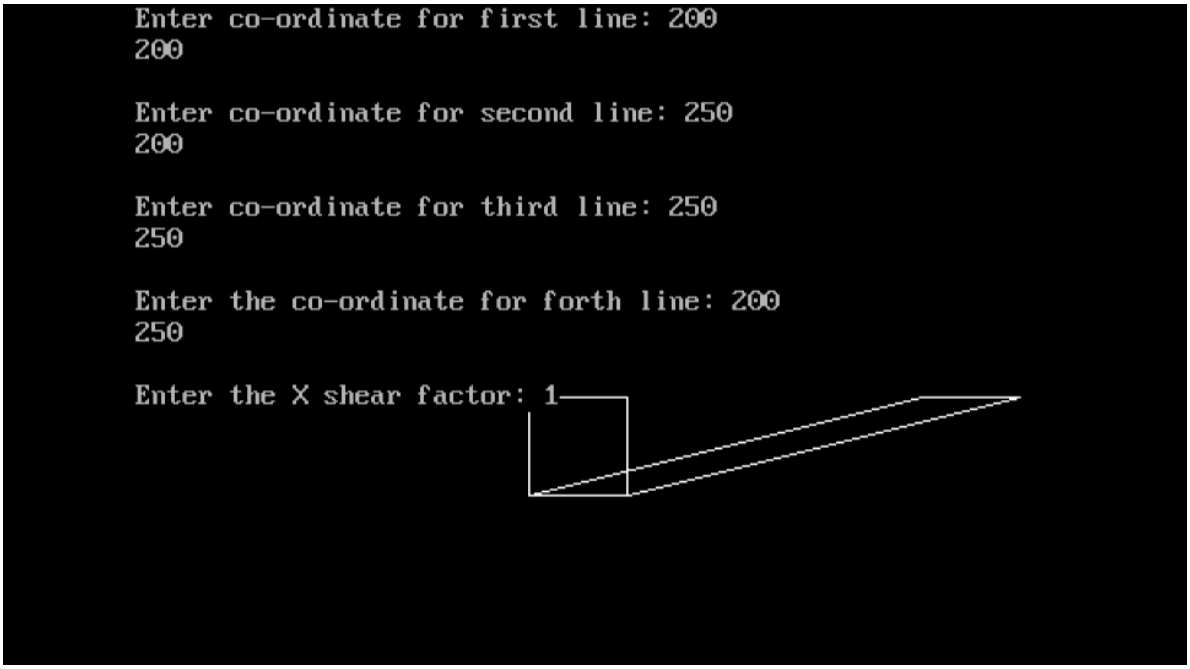
Practical No. 7.2

Aim:- Write a graphics program for X-shear translation.

Source code:-

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<math.h>
#include<dos.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
int x1,y1,x2,y2,x3,y3,x4,y4;
cout<<"Enter co-ordinate for first line: ";
cin>>x1>>y1;
cout<<"\nEnter co-ordinate for second line: ";
cin>>x2>>y2;
cout<<"\nEnter co-ordinate for third line: ";
cin>>x3>>y3;
cout<<"\nEnter the co-ordinate for forth line: ";
cin>>x4>>y4;
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x4,y4);
line(x4,y4,x1,y1);
int shx;
cout<<"\nEnter the X shear factor: ";
cin>>shx;
x1=x1+shx*y1;
x2=x2+shx*y2;
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x4,y4);
line(x4,y4,x1,y1);
getch();
}
```

Output:-


Enter co-ordinate for first line: 200
200
Enter co-ordinate for second line: 250
200
Enter co-ordinate for third line: 250
250
Enter the co-ordinate for forth line: 200
250
Enter the X shear factor: 1

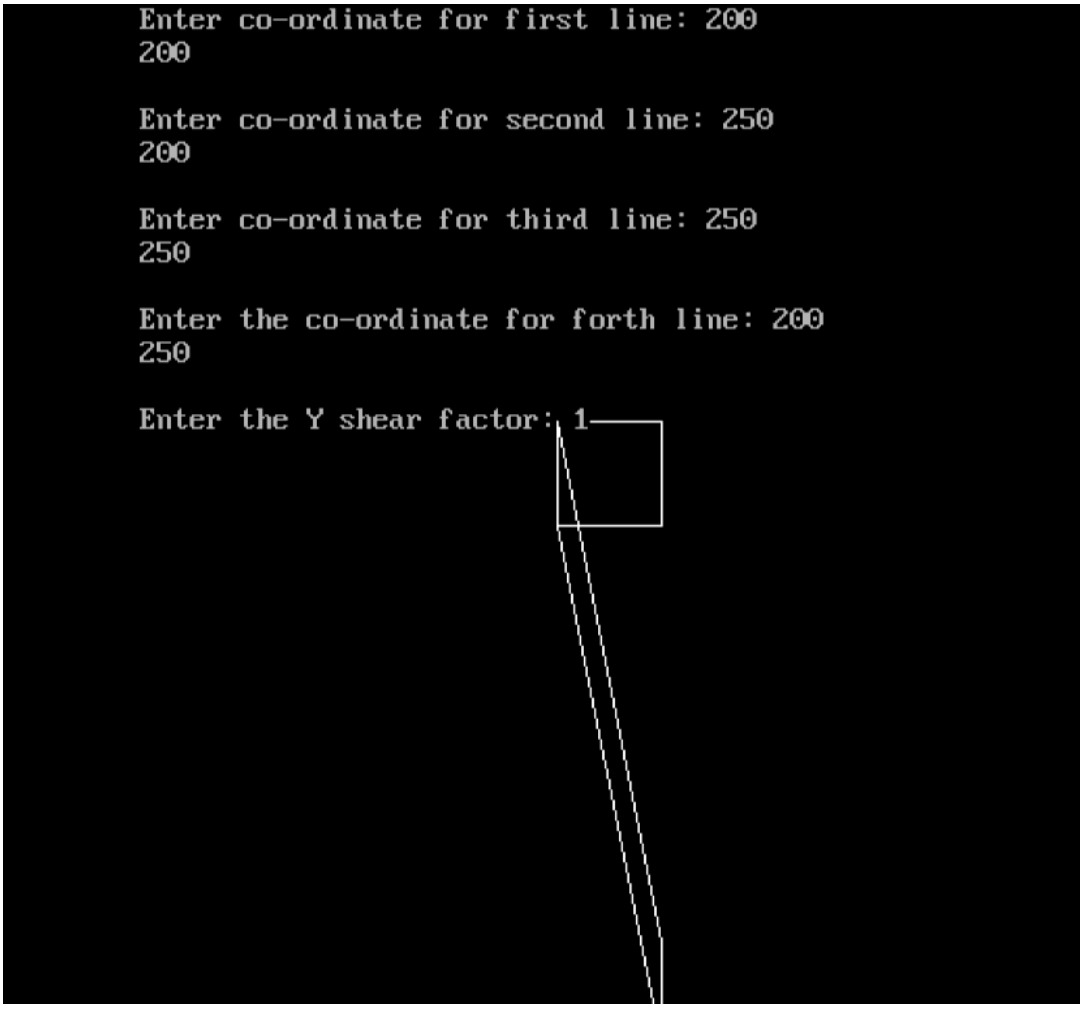
Practical No. 7.2

Aim:- Write a graphics program for Y-shear translation.

Source code:-

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<math.h>
#include<dos.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
int x1,y1,x2,y2,x3,y3,x4,y4;
cout<<"Enter co-ordinate for first line: ";
cin>>x1>>y1;
cout<<"\nEnter co-ordinate for second line: ";
cin>>x2>>y2;
cout<<"\nEnter co-ordinate for third line: ";
cin>>x3>>y3;
cout<<"\nEnter the co-ordinate for forth line: ";
cin>>x4>>y4;
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x4,y4);
line(x4,y4,x1,y1);
int shy;
cout<<"\nEnter the Y shear factor: ";
cin>>shy;
y2=y2+shy*x2;
y3=y3+shy*x3;
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x4,y4);
line(x4,y4,x1,y1);
getch();
}
```

Output:-


Enter co-ordinate for first line: 200
200
Enter co-ordinate for second line: 250
200
Enter co-ordinate for third line: 250
250
Enter the co-ordinate for forth line: 200
250
Enter the Y shear factor: 1

Practical No. 7.2

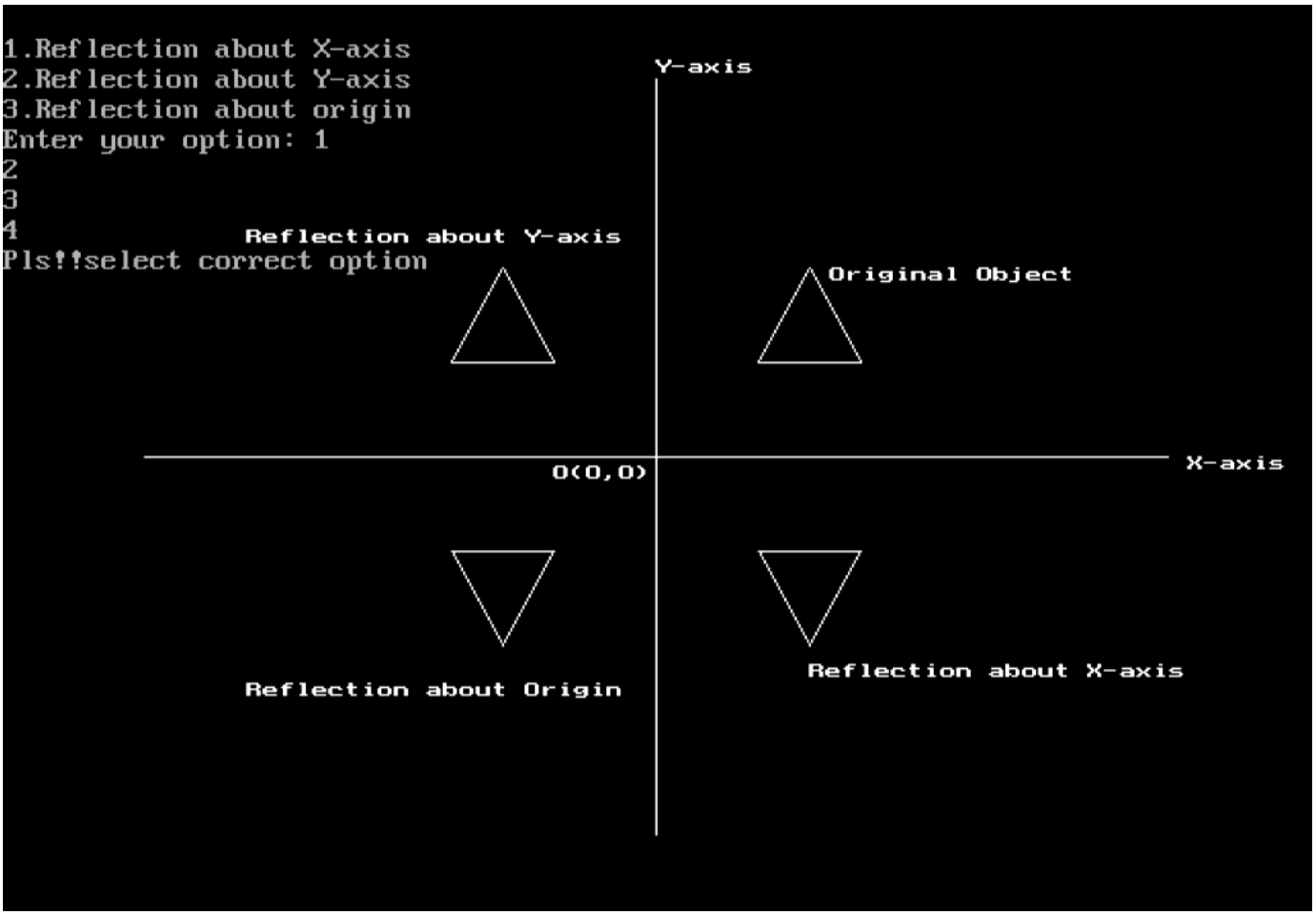
Aim: Write a graphics program for reflection.

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
#include<stdlib.h>
void main()
{
int gd,gm;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
int xmid,ymid;
xmid=getmaxx()/2;
ymid=getmaxy()/2;
line(xmid-250,ymid,xmid+250,ymid);
outtextxy(xmid+260,ymid,"X-axis");
line(xmid,ymid-200,xmid,ymid+200);
outtextxy(xmid,ymid-210,"Y-axis");
outtextxy(xmid-50,ymid+5,"O(0,0)");
line(xmid+50,ymid-50,xmid+100,ymid-50);//bottom line
line(xmid+100,ymid-50,xmid+75,ymid-100);//reight line
line(xmid+50,ymid-50,xmid+75,ymid-100);//left line
outtextxy(xmid+85,ymid-100,"Original Object");
int op;
cout<<"\n1.Reflection about X-axis";
cout<<"\n2.Reflection about Y-axis";
cout<<"\n3.Reflection about origin";
cout<<"\nEnter your option: ";
do
{
cin>>op;
switch(op)
{
case 1:
line(xmid+50,ymid+50,xmid+100,ymid+50);//bottom line
line(xmid+100,ymid+50,xmid+75,ymid+100);//reight line
line(xmid+75,ymid+100,xmid+50,ymid+50);//left line
outtextxy(xmid+75,ymid+110,"Reflection about X-axis");
break;
case 2:
line(xmid-50,ymid-50,xmid-100,ymid-50);//bottom line
line(xmid-100,ymid-50,xmid-75,ymid-100);//reight line
line(xmid-75,ymid-100,xmid-50,ymid-50);//left line
outtextxy(xmid-200,ymid-120,"Reflection about Y-axis");
break;
case 3:
line(xmid-50,ymid+50,xmid-100,ymid+50);//bottom line
line(xmid-100,ymid+50,xmid-75,ymid+100);//reight line
line(xmid-50,ymid+50,xmid-75,ymid+100);//left line
```

```
outtextxy(xmid-200,ymid+120,"Reflection about Origin");
break;
default:
cout<<"Pls!!select correct option";
break;
}
}while(op<4);
getch();
}
```

Output:



Practical no. 9.2

Aim:- Implementation of Translation using 3-D object(only coordinate calculation).

Source Code:

```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
void main()
{
int n,i;
int a[20][3],b[20][3],xi[20];
float tx,ty,tz;
clrscr();
cout<<"\n\n\tEnter the no. of edges of polygon : ";
cin>>n;
cout<<"\n\n\tEnter the cordinates of polygon :\n\n\n ";
for(i=0;i<n;i++) {
cin>>a[i][0]>>a[i][1]>>a[i][2];
}
cout<<"Enter Translating parameters :\n";
cin>>tx>>ty>>tz;
for(i=0;i<n;i++) {
b[i][0] = (int) a[i][0] + tx;
b[i][1] = (int) a[i][1] + ty;
b[i][2] = (int) a[i][2] + tz;
}
cout<<"\n 3D-Translation \n";
for(i=0; i<n ; i++) {
cout<<"\n "<<a[i][0]<<"\t"<<a[i][1]<<"\t"<<a[i][2]<<"\t\t"
<<b[i][0]<<"\t"<<b[i][1]<<"\t"<<b[i][2];
}
getch();
}
```

Output:

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

        Enter the cordinates of polygon :

 1 2 3
 3 6 8
 2 1 5
Enter Translating parameters :
2 5 8

 3D-Translation

 1      2      3              3      7      11
 3      6      8              5      11     16
 2      1      5              4      6      13
```

Practical No. 9.2

Aim:- Implementation of scaling in 3D object(Only coordinates calculation).

Source Code:-

```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
main()
{
int n,i;
int a[20][3],b[20][3],xi[20];
float sx,sy,sz;
clrscr();
cout<<"\n\n\tEnter the no. of edges of polygon : ";
cin>>n;
cout<<"\n\n\tEnter the cordinates of polygon :\n\n\n ";
for(i=0;i<n;i++){
cin>>a[i][0]>>a[i][1]>>a[i][2];
}
cout<<"Enter scaling parameters :\n";
cin>>sx>>sy>>sz;
for(i=0;i<n;i++){
b[i][0] = (int) a[i][0] * sx;
b[i][1] = (int) a[i][1] * sy;
b[i][2] = (int) a[i][2] * sz;
}
cout<<"\n 3D-Scaling \n";
for(i=0; i<n ; i++) {
cout<<"\n "<<a[i][0]<<"\t"<<a[i][1]<<"\t"<<a[i][2]<<"\t\t"
<<b[i][0]<<"\t"<<b[i][1]<<"\t"<<b[i][2];
}
getch();
}
```

Output:

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

        Enter the cordinates of polygon :

 2 3 5
 4 7 9
 7 4 8
Enter Scaling parameters :
3 5 7

 3D-Scaling

 2      3      5              6      15     35
 4      7      9              12     35     63
 7      4      8              21     20     56
```

Practical No. 9.2

Aim:- Implementation of rotation in 3D object(Only coordinates calculation).

Source Code:-

```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <process.h>
void main()
{
int n,i,ch;
int a[20][3],b[20][3];
float theta;
clrscr();
cout<<"\n\n\tEnter the no. of edges of polygon : ";
cin>>n;
cout<<"\n\n\tEnter the cordinates of polygon :\n\n\n ";
for(i=0;i<n;i++) {
cin>>a[i][0]>>a[i][1]>>a[i][2];
}
cout<<"\nEnter angle of rotation : ";
cin>>theta;
theta = (theta * 3.14)/180;
cout<<"\n1.Rotate About X Axis";
cout<<"\n2.Rotate About Y Axis";
cout<<"\n3.Rotate About Z Axis";
cout<<"\n4.Exit";
cout<<"\nEnter your choice: ";
cin>>ch;
switch(ch)
{
case 3:
for(i=0;i<n;i++) {
b[i][0] = (int) (a[i][0] * cos(theta)-(a[i][1]*sin(theta)));
b[i][1] = (int) (a[i][0] * sin(theta)+(a[i][1]*cos(theta)));
b[i][2] = a[i][2];
}
break;
case 1:
for(i=0;i<n;i++) {
b[i][0] = a[i][0];
b[i][1] = (int) (a[i][1] * cos(theta)-(a[i][2]*sin(theta)));
b[i][2] = (int) (a[i][1] * sin(theta)+(a[i][2]*cos(theta)));
}
break;
case 2:
for(i=0;i<n;i++) {
b[i][0] = (int) (a[i][2] * sin(theta)+(a[i][2]*cos(theta)));
b[i][1] = a[i][1];
b[i][2] = (int) (a[i][2] * cos(theta)-(a[i][0]*sin(theta)));
}
break;
case 4:
```

```
exit(0);
break;
}
for(i=0;i<n;i++) {
cout<<"\n "<<a[i][0]<<"\t"<<a[i][1]<<"\t"<<a[i][2]<<"\t\t"
<<b[i][0]<<"\t"<<b[i][1]<<"\t"<<b[i][2];
}
getch();
}
```

Output:

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

        Enter the cordinates of polygon :

 3 5 7
 2 4 6
 1 3 5

Enter angle of rotation : 90

1.Rotate About X Axis
2.Rotate About Y Axis
3.Rotate About Z Axis
4.Exit
Enter your choice: 2

 3      5      7              7      5      -2
 2      4      6              6      4      -1
 1      3      5              5      3      0
```

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

        Enter the cordinates of polygon :

 3 5 7
 2 4 6
 1 3 5

Enter angle of rotation : 90

1.Rotate About X Axis
2.Rotate About Y Axis
3.Rotate About Z Axis
4.Exit
Enter your choice: 3

 3      5      7              -4     3      7
 2      4      6              -3     2      6
 1      3      5              -2     1      5_
```

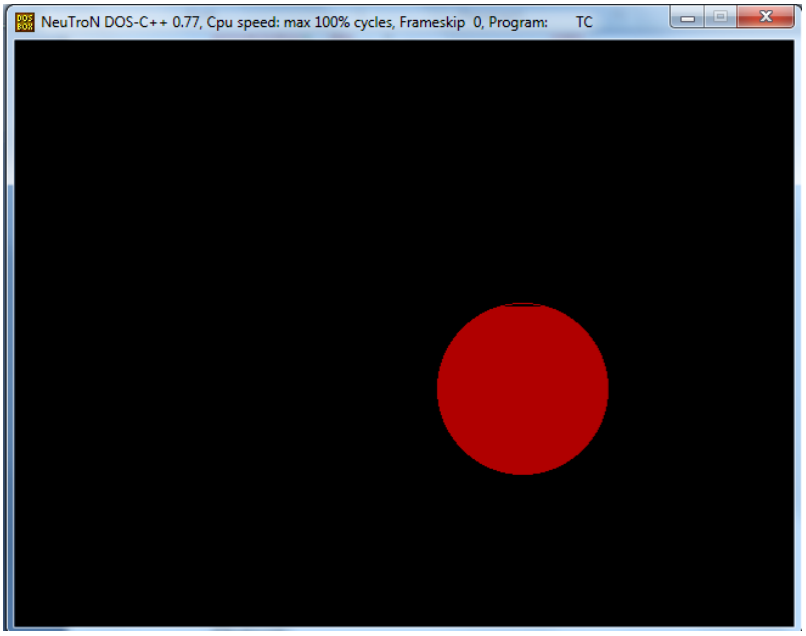
Practical No. 11.1

Aim:- Implementation of Animation(Bouncing Ball).

Source Code:-

```
#include <graphics.h>
#include <conio.h>
#include<dos.h>
int main()
{
int gd = DETECT, gm, i;
initgraph(&gd, &gm, "c:\\turboc3\\bgi");
int x,y=0,j,t=400,c=1;
setcolor(RED);
setfillstyle(SOLID_FILL,RED);
for(x=40;x<602;x++)
{
cleardevice();
circle(x,y,70);
floodfill(x,y,RED);
delay(40);
if(y>=400)
{
c=0;
t-=20;
}
if(y<=(400-t))
c=1;
y=y+(c?15:-15);
}
getch();
}
```

Output:



Practical No. 11.2

Aim:- Implementation of Digital clock.

Source Code:

```
#include <conio.h>
#include <graphics.h>
#include <time.h>
#include <dos.h>
#include <string.h>

int main() {
   int gd = DETECT, gm;
   int midx, midy;
   long current_time;
   char timeStr[256];

   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

   /* mid pixel in horizontal and vertical axis */
   midx = getmaxx() / 2;
   midy = getmaxy() / 2;

   while (!kbhit()) {
      cleardevice();
      setcolor(WHITE);
      setfillstyle(SOLID_FILL, WHITE);
rectangle(midx - 250, midy - 40, midx + 250, midy + 40);
      floodfill(midx, midy, WHITE);
      /* Get Current epoch time in seconds */
      current_time = time(NULL);
      /* store the date and time in string */
      strcpy(timeStr, ctime(&current_time));
      setcolor(RED);
      settextjustify(CENTER_TEXT, CENTER_TEXT);
      settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 4);

      moveto(midx, midy);
      /* print current time */
      outtext(timeStr);
      /* Add delay of 1000 milliseconds(1 second) */
      delay(1000);
   }

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

Output:-