**1.Write a Program to show the use of inbuilt functions in graphics.**

#include<graphics.h>

#include<iostream.h>

#include<conio.h>

#include<stdlib.h>

#include<dos.h>

class graphics //create class

{

public:

void hut(); //function declaration

void smiley(); //function declaration

void car(); //function declaration

};

void graphics::hut() //function definition

{

line(100,100,150,50);

line(150,50,200,100);

line(150,50,350,50);

line(350,50,400,100);

rectangle(100,100,200,200);

rectangle(200,100,400,200);

rectangle(130,130,170,200); //for door

rectangle(250,120,350,180); //for window

setfillstyle(SOLID\_FILL,BROWN);

floodfill(131,131,WHITE);

floodfill(201,101,WHITE);

floodfill(170,190,WHITE);

setfillstyle(HATCH\_FILL,GREEN);//fillstyle pattern

floodfill(101,101,WHITE);

floodfill(150,52,WHITE);

floodfill(163,55,WHITE);

floodfill(251,121,WHITE);

floodfill(160,99,WHITE);

}

void graphics::smiley() //function definition

{

setcolor(YELLOW);

circle(295,250,60);

setfillstyle(SOLID\_FILL,YELLOW);

floodfill(295,250,YELLOW); //fill color

setcolor(BLACK);

setfillstyle(SOLID\_FILL,BLACK);

fillellipse(270,235,3,3); //for eyes

fillellipse(315,235,3,3); //for eyes

arc(295,270,190,350,20); //for smile

}

void graphics::car() //function definition

{

int i;

for(i=0;i<=120;i++)

{

rectangle(300+i,100,30+i,250);

rectangle(210+i,130,260+i,180); //for window

circle(230+i,250,20); //for wheel

circle(100+i,250,20); //for wheel

delay(50); //time for moving car

cleardevice();

}

}

void main()

{

int ch;

graphics g; //create object

clrscr();

int gd=DETECT,gm;

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

while(1) //conditional loop

{

cout<<"Enter 1 for Smiley"<<endl;

cout<<"Enter 2 for hut"<<endl;

cout<<"Enter 3 for car"<<endl;

cout<<"Enter 4 for exit"<<endl;

cout<<"Enter choice"<<endl;

cin>>ch; //enter choice

switch(ch) //switch case

{

case 1:

clrscr();

g.smiley(); //function call

break;

case 2:

clrscr();

g.hut(); //function call

break;

case 3:

clrscr();

g.car(); //function call

break;

case 4:

exit(0);

default:

cout<<"WRONG VALUE"<<endl;

}

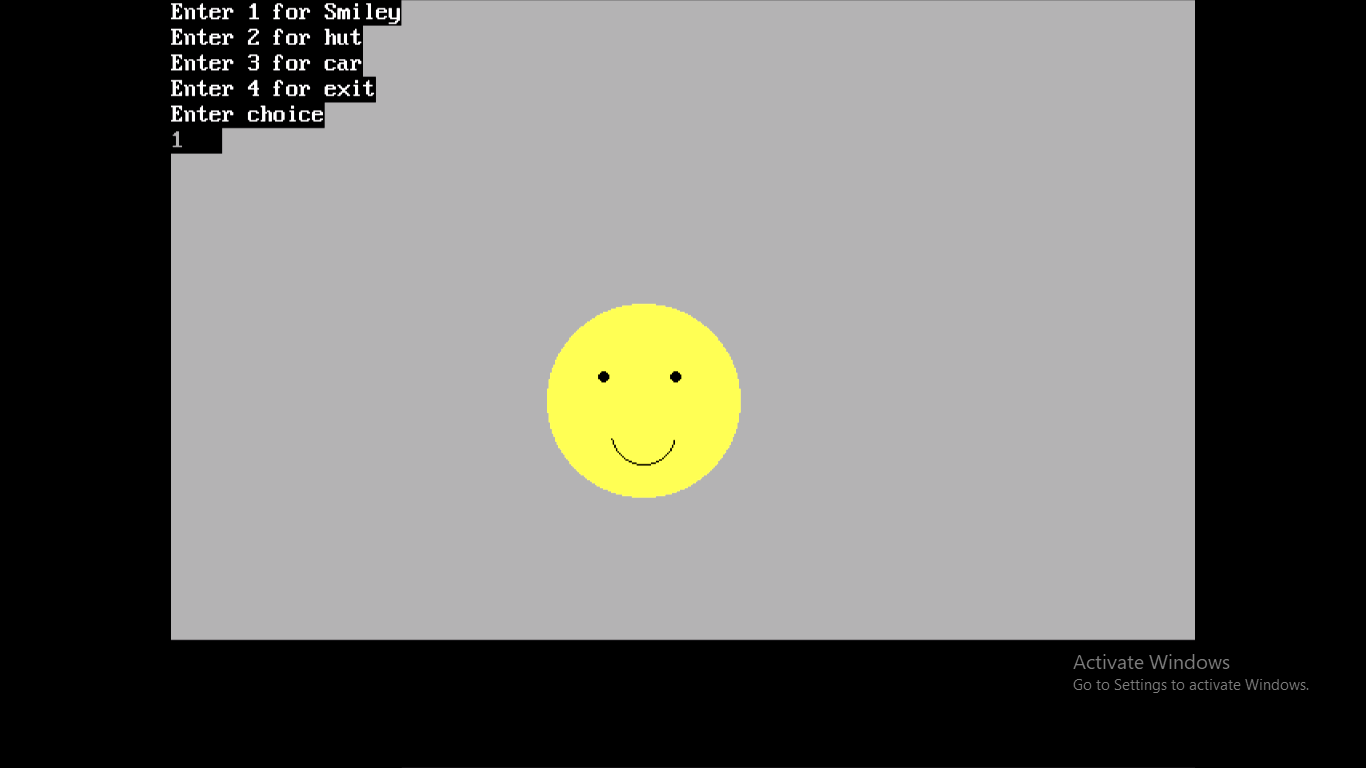
}

getch();

closegraph();

}

**OUTPUT**





**2.Write a Program to draw a line using direct method in graphics.**

include<graphics.h>

#include<iostream.h>

#include<conio.h>

class direct //create class

{

public:

void getdisp(); //function declaration

};

void direct::getdisp() //function definition

{

int x,y,x1,x2,y1,y2,b,m,dy,dx;

cout<<"Enter the first endpoints(x1,y1)"<<endl;

cin>>x1>>y1; //enter first endpoints

cout<<"Enter the second endpoints(x2,y2)"<<endl;

cin>>x2>>y2; //enter second endpoints

dx=(x2-x1);

dy=(y2-y1);

m=dy/dx; //find slope

b=y1-(m\*x1);

x=x1;

y=y1;

putpixel(x,y,RED); //plot points

while(x<=x2) //conditional loop

{

if(m<=1) //check condition

{

x=x+1;

y=((m\*x)+b);

putpixel(x,y,GREEN); //plot points

}

else

{

y=y+1;

x=(y-b)/m;

putpixel(x,y,WHITE); //plot points

}

}

}

void main()

{

clrscr();

int gd=DETECT,gm;

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

direct d; //create object

d.getdisp(); //function call

getch();

closegraph();

}

**OUTPUT**

****

**3.Write a menu driven Program of line using dda,bresenham’s methods in graphics.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<stdlib.h>

class Line //create class

{

public:

void DDA(); //function declaration

void BRESENHAM(); //function declaration

};

void Line::DDA() //function definition

{

int x,y,x1,x2,y1,y2,m,dy,dx,i,xinc,yinc,length;

cout<<"Enter the first endpoints (x1,y1)"<<endl;

cin>>x1>>y1; //input first endpoints

cout<<"Enter the second endpoints (x2,y2)"<<endl;

cin>>x2>>y2; //input second endpoints

dx=(x2-x1);

dy=(y2-y1);

x=x1;

y=y1;

if (dx>=dy)

{

length=dx;

}

else

{

length=dy;

}

xinc=dx/length;

yinc=dy/length;

putpixel(x,y,BLUE); //plot the points

for(i=1;i<=length;i++)

{

x=x+xinc;

y=y+yinc;

putpixel(x,y,RED); //plot the points

}

}

void Line::BRESENHAM() //function definition

{

int x,y,x1,y1,x2,y2,d,ds,dt,xend,dx,dy;

cout<<"Enter the first endpoints (x1,y1)"<<endl;

cin>>x1>>y1; //input first endpoints

cout<<"Enter the second endpoints (x2,y2)"<<endl;

cin>>x2>>y2; //input second endpoints

dx=(x2-x1);

dy=(y2-y1);

d=2\*dy-dx;

ds=2\*dy;

dt=2\*(dy-dx);

if(dx>0) //check condition

{

x=x1;

y=y1;

xend=x2;

}

else

{

x=x2;

y=y2;

xend=x1;

}

putpixel(x,y,BLUE); //plot the points

while(x<xend) //conditional loop

{

x=x+1;

if(d<0) //check condition

{

d=d+ds;

}

else

{

d=d+dt;

y=y+1;

}

putpixel(x,y,YELLOW); //plot the points

}

}

void main()

{ clrscr();

int ch;

Line l; //create object

int gd=DETECT,gm;

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

while(1) //conditional loop

{

cout<<"Enter 1 for DDA method"<<endl;

cout<<"Enter 2 for BRESENHAM'S method"<<endl;

cout<<"Enter 3 for exit"<<endl;

cout<<"Enter choice"<<endl;

cin>>ch; //enter choice

switch(ch) //switch case

{

case 1:

l.DDA(); //function call

break;

case 2:

l.BRESENHAM(); //function call

break;

case 3:

exit(0);

default:

cout<<"WRONG VALUE"<<endl;

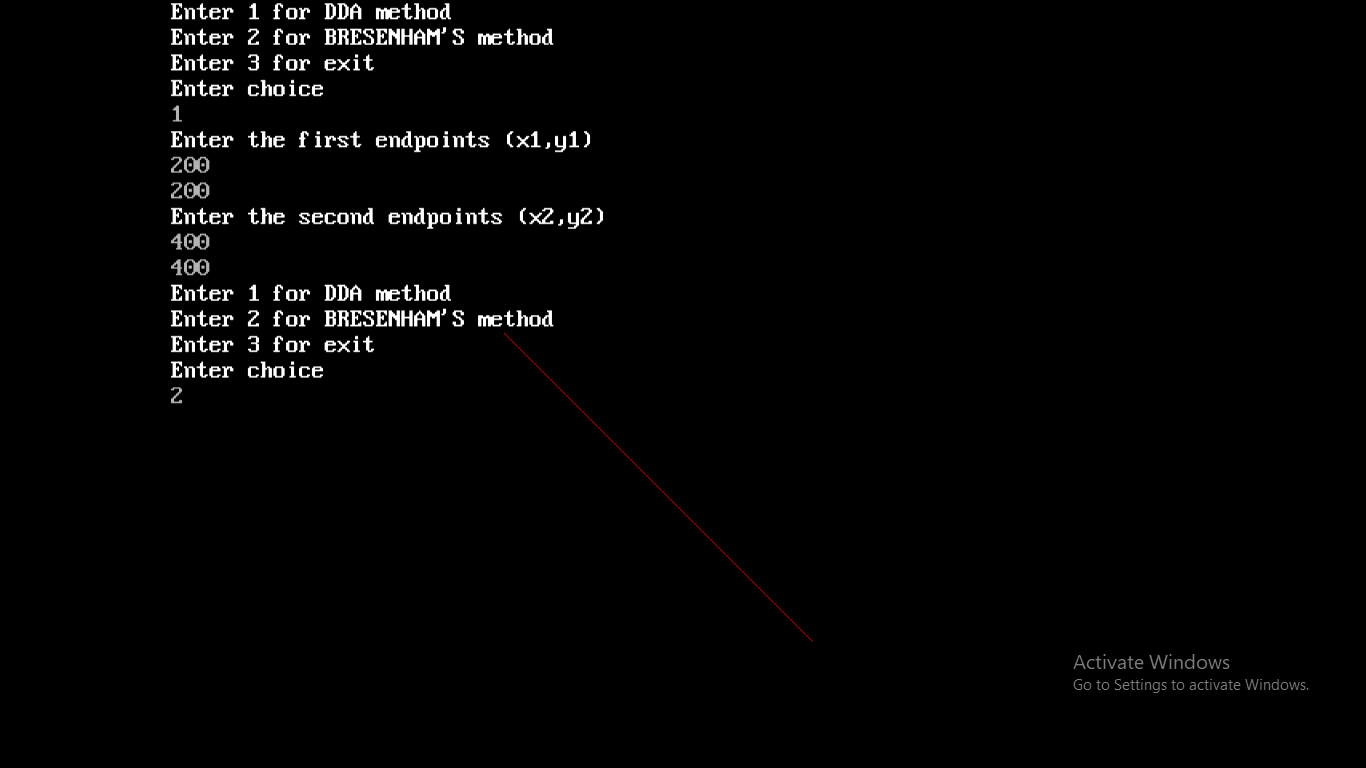
}

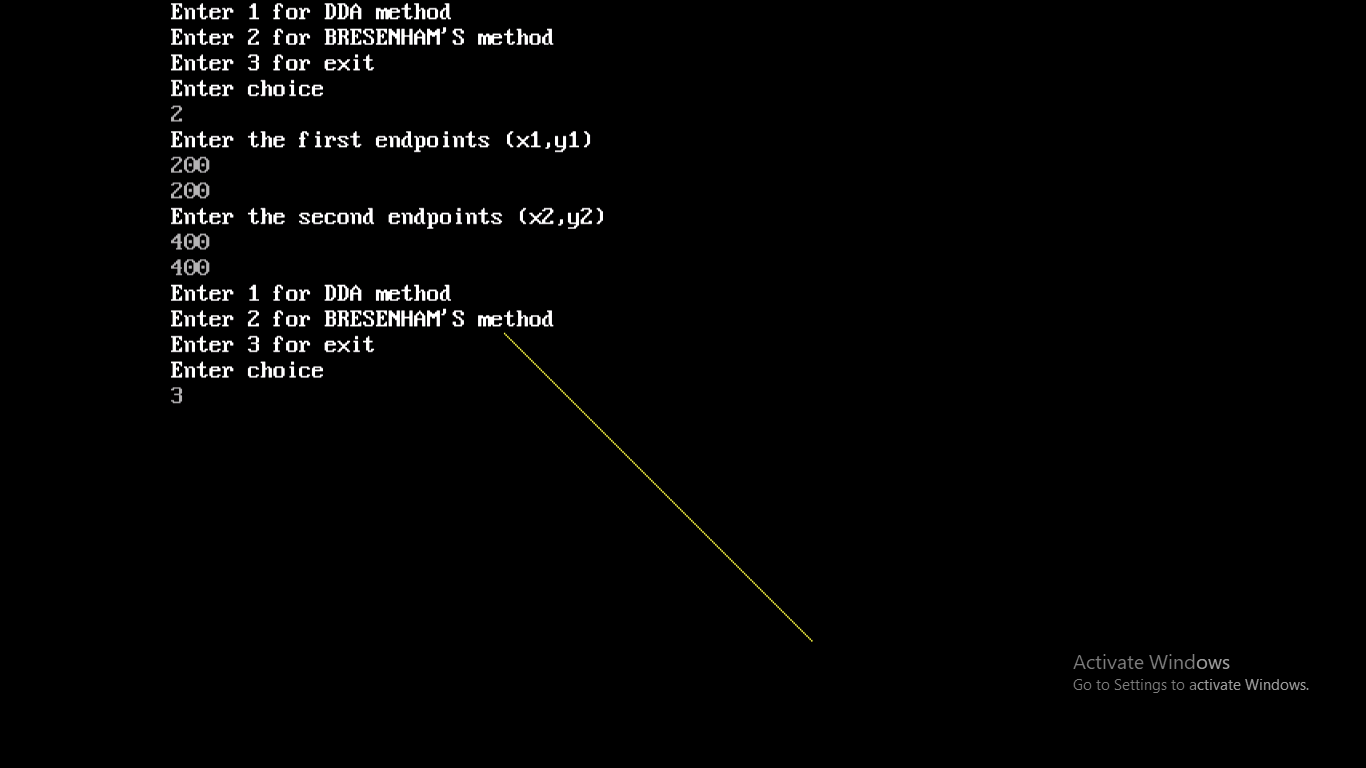
}

getch();

}

**OUTPUT**

****

****

**4.Write a menu driven Program of circle using polynomial,bresenham’s,mid-point methods in graphics.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

#include<stdlib.h>

class Circle //create class

{

public:

void direct(); //function declaration

void bresenham(); //function declaration

void midpoint(); //function declaration

};

void Circle::direct() //function definition

{

int h,k,x,r;

double xend,y;

int i;

cout<<"Enter the center point of the circle"<<endl;

cin>>h>>k; //input center points

cout<<"Enter the radius of the circle"<<endl;

cin>>r; //input radius

x=0;

xend=r/sqrt(2);

while(x<=xend) //conditional loop

{

y=sqrt(r\*r-x\*x);

//plot eight points

putpixel(x+h,y+k,5);

putpixel(y+h,x+k,5);

putpixel(-y+h,x+k,5);

putpixel(-x+h,y+k,5);

putpixel(-x+h,-y+k,5);

putpixel(-y+h,-x+k,5);

putpixel(y+h,-x+k,5);

putpixel(x+h,-y+k,5);

x=x+1;

}

}

void Circle::bresenham() //function definition

{

int h,k,x,y,r,d;

cout<<"Enter the center point of the circle"<<endl;

cin>>h>>k; //input center points

cout<<"Enter the radius of the circle"<<endl;

cin>>r; //input radius

x=0;

y=r;

d=3-2\*r;

while(x<=y) //conditional loop

{

putpixel(x+h,y+k,4); //plot eight points

putpixel(y+h,x+k,4);

putpixel(-y+h,x+k,4);

putpixel(-x+h,y+k,4);

putpixel(-x+h,-y+k,4);

putpixel(-y+h,-x+k,4);

putpixel(y+h,-x+k,4);

putpixel(x+h,-y+k,4);

x++;

if(d<0) //check condition

{

d=d+4\*x+6;

}

else

{

y--;

d=d+4\*(x-y)+10;

}

putpixel(x+h,y+k,6); //plot eight points

putpixel(y+h,x+k,6);

putpixel(-y+h,x+k,6);

putpixel(-x+h,y+k,6);

putpixel(-x+h,-y+k,6);

putpixel(-y+h,-x+k,6);

putpixel(y+h,-x+k,6);

putpixel(x+h,-y+k,6);

}

}

void Circle::midpoint() //function definition

{

int h,k,x,y,r,p;

cout<<"Enter the center point of the circle"<<endl;

cin>>h>>k; //input center points

cout<<"Enter the radius of the circle"<<endl;

cin>>r; //input radius

x=0;

y=r;

p=1-r;

while(x<=y) //conditional loop

{

//plot eight points

putpixel(x+h,y+k,9);

putpixel(y+h,x+k,9);

putpixel(-y+h,x+k,9);

putpixel(-x+h,y+k,9);

putpixel(-x+h,-y+k,9);

putpixel(-y+h,-x+k,9);

putpixel(y+h,-x+k,9);

putpixel(x+h,-y+k,9);

x++;

if(p<0) //check condition

{

p=p+2\*x+3;

}

else

{

y--;

p=p+2\*(x-y)+5;

}

//plot eight points

putpixel(x+h,y+k,8);

putpixel(y+h,x+k,8);

putpixel(-y+h,x+k,8);

putpixel(-x+h,y+k,8);

putpixel(-x+h,-y+k,8);

putpixel(-y+h,-x+k,8);

putpixel(y+h,-x+k,8);

putpixel(x+h,-y+k,8);

}

}

void main()

{

clrscr();

int ch;

Circle c; //create object

int gd=DETECT,gm;

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

while(1) //conditional loop

{

cout<<"Enter 1 for Direct method"<<endl;

cout<<"Enter 2 for Bresenham's method"<<endl;

cout<<"Enter 3 for Mid point method"<<endl;

cout<<"Enter 4 for exit"<<endl;

cout<<"Enter choice"<<endl;

cin>>ch; //enter choice

switch(ch) //switch case

{

case 1:

c.direct(); //function call

break;

case 2:

c.bresenham(); //function call

break;

case 3:

c.midpoint();

break;

case 4:

exit(0);

default:

cout<<"WRONG VALUE"<<endl;

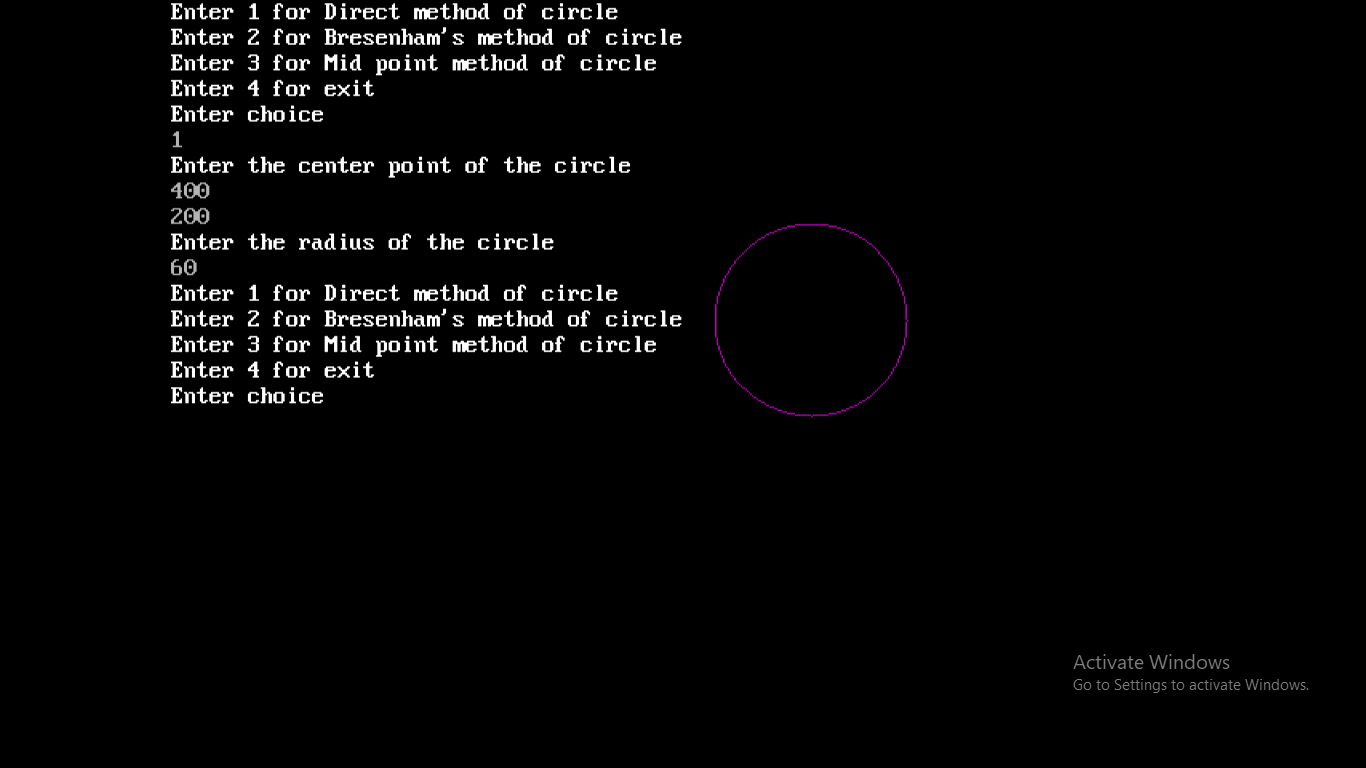
}

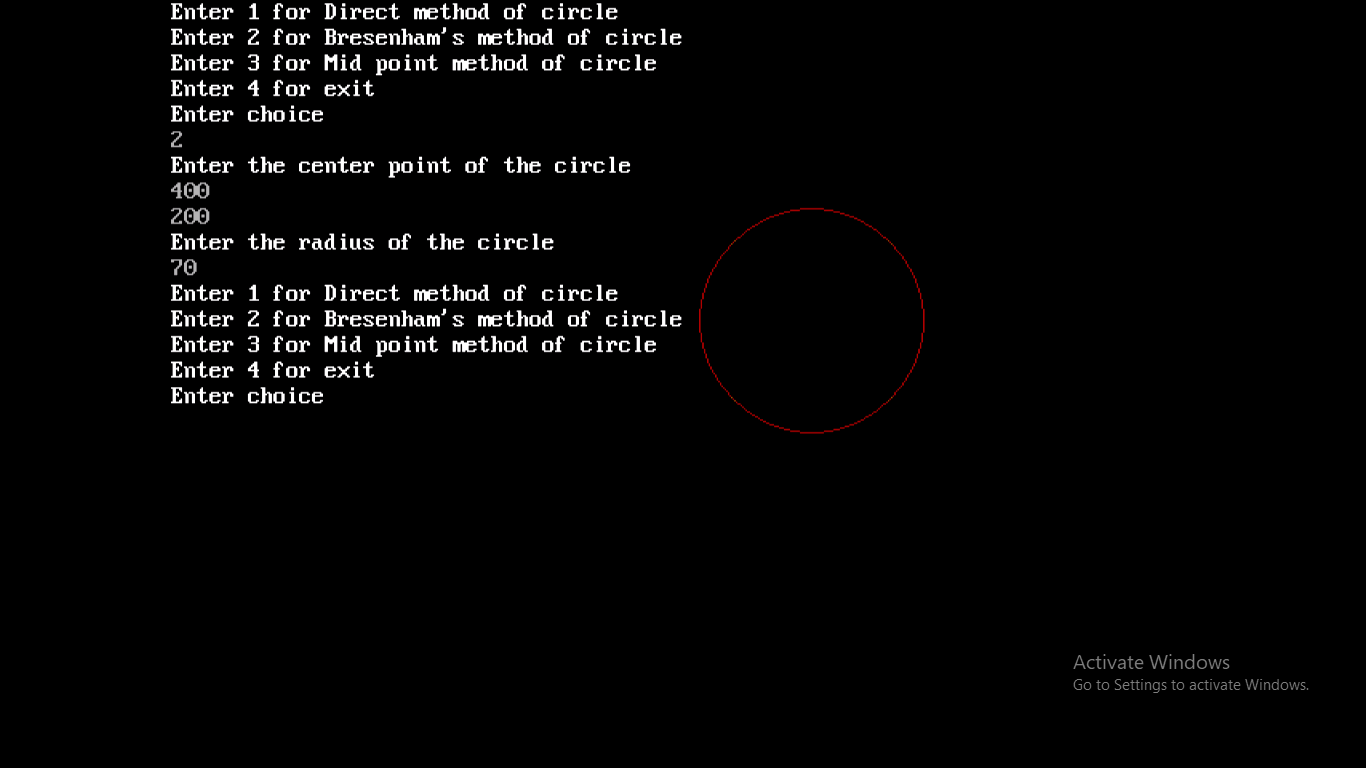
}

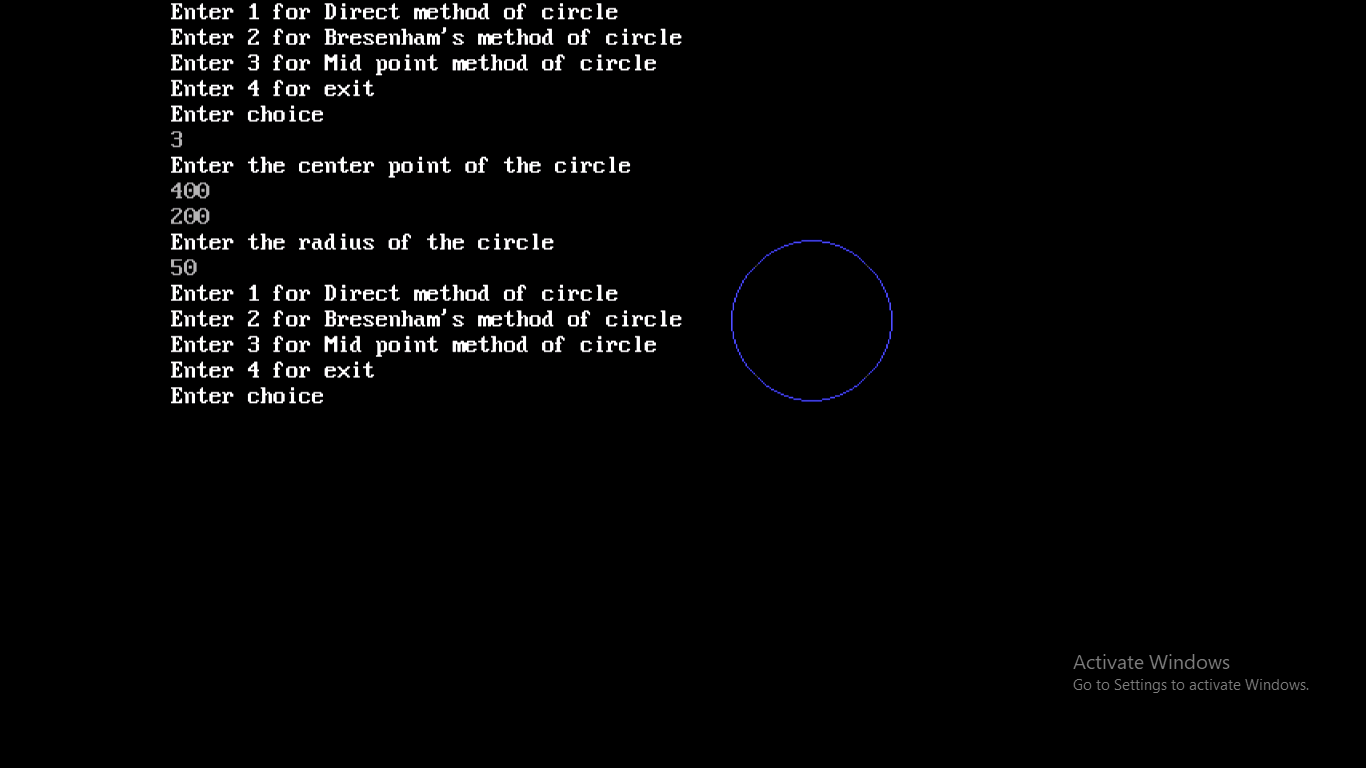
getch();

}

**OUTPUT**

****

****

****

**5.Write a Program of Area filling using boundaryfill and floodfill methods in graphics.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<stdlib.h>

class FillArea //class declaration

{

public:

void boundary(int,int,int,int); //function declaration

void floodfill2(int,int,int,int);

};

void FillArea::boundary(int seed\_x,int seed\_y,int

fillcolor,int boundarycolor) //function definition

{

int current;

current=getpixel(seed\_x,seed\_y);

if((current!=boundarycolor)&&(current!=fillcolor))

{

putpixel(seed\_x,seed\_y,fillcolor); //plot points

//function call with passing parameters

boundary(seed\_x+1,seed\_y,fillcolor,boundarycolor);

boundary(seed\_x-1,seed\_y,fillcolor,boundarycolor);

boundary(seed\_x,seed\_y+1,fillcolor,boundarycolor);

boundary(seed\_x,seed\_y-1,fillcolor,boundarycolor);

boundary(seed\_x+1,seed\_y,fillcolor,boundarycolor);

boundary(seed\_x,seed\_y+1,fillcolor,boundarycolor);

boundary(seed\_x-1,seed\_y,fillcolor,boundarycolor);

boundary(seed\_x,seed\_y-1,fillcolor,boundarycolor);

}

}

void FillArea::floodfill2(int seed\_x,int seed\_y,int new\_color,int old\_color)

{

int current;

current=getpixel(seed\_x,seed\_y);

if (current==old\_color)

{

putpixel(seed\_x,seed\_y, new\_color); //plot points

//function call with passing parameters

floodfill2(seed\_x+1, seed\_y,new\_color,old\_color);

floodfill2(seed\_x-1 ,seed\_y,new\_color,old\_color);

floodfill2(seed\_x ,seed\_y+1,new\_color,old\_color);

floodfill2(seed\_x ,seed\_y-1,new\_color,old\_color);

floodfill2(seed\_x-1,seed\_y-1,new\_color,old\_color);

floodfill2(seed\_x+1,seed\_y+1,new\_color,old\_color);

floodfill2(seed\_x-1,seed\_y+1,new\_color,old\_color);

floodfill2(seed\_x+1,seed\_y-1,new\_color,old\_color);

}

}

void main()

{

clrscr();

int ch,l,t,r,b,x,y;

FillArea fa; //create object

clrscr();

int gm,gd=DETECT;

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

while(1) //while loop

{

cout<<"enter 1 for boundary fill Algorithm"<<endl;

cout<<"enter 2 for flood fill Algorithm "<<endl;

cout<<"enter 3 for exit"<<endl;

cout<<"enter choice"<<endl;

cin>>ch; //enter choice

switch(ch) //switch case

{

case 1:

cout<<"enter the value of rectangle ";

cin>>l>>t>>r>>b;

cout<<"enter the value of seed point";

cin>>x>>y;

rectangle(l,t,r,b);

fa.boundary(x,y,4,15); //function call

break;

case 2:

cout<<"enter the value of rectangle ";

cin>>l>>t>>r>>b;

cout<<"enter the value of seed point";

cin>>x>>y;

rectangle(l,t,r,b);

fa.floodfill2(x,y,12,0); //function call

break;

case 3:

exit(0);

default:

cout<<"wrong value"<<endl;

}

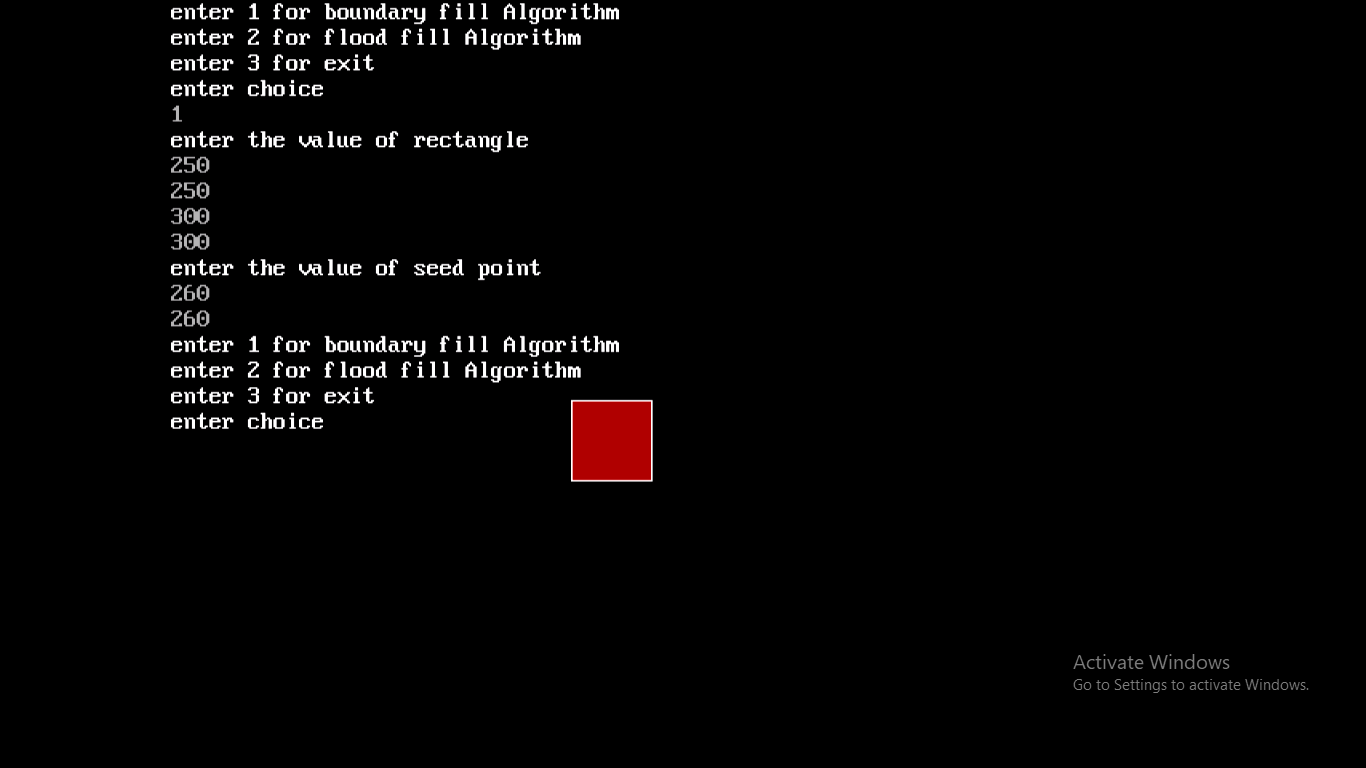
}

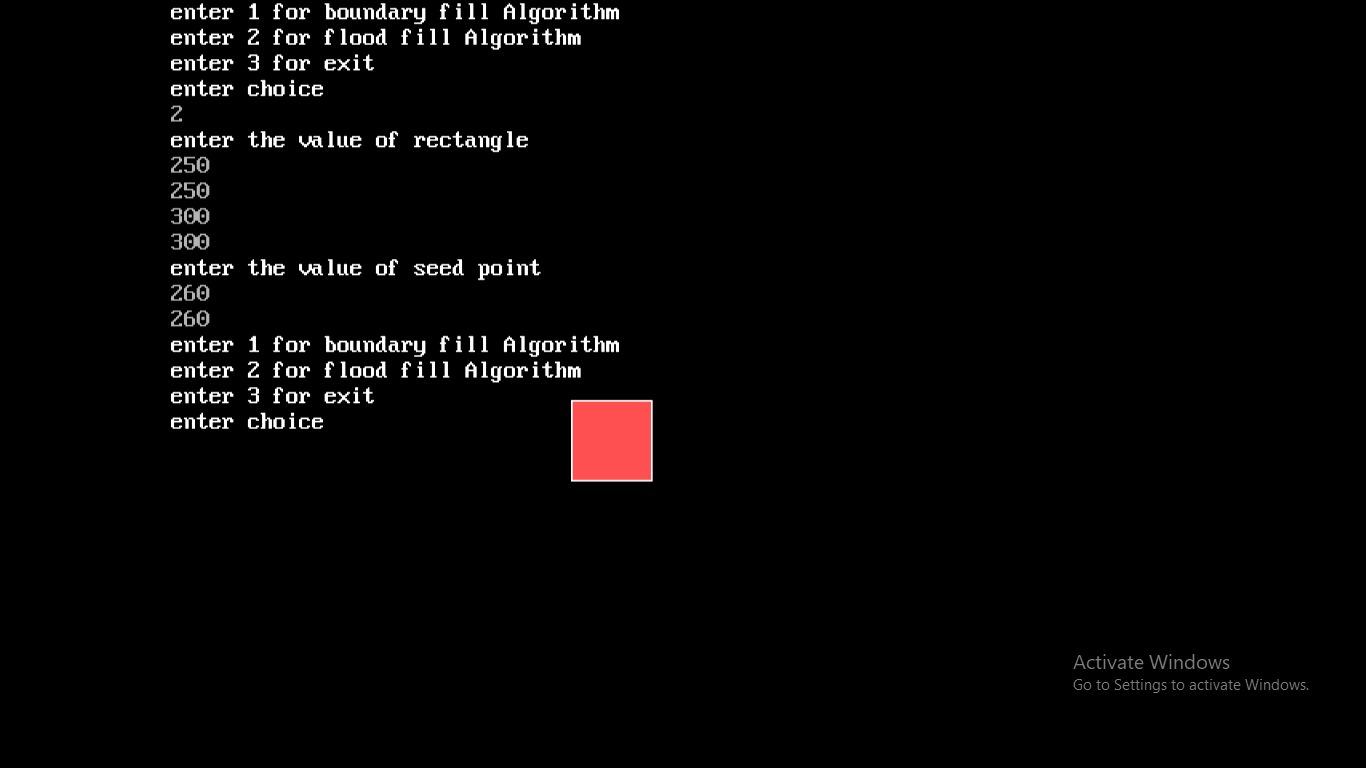
getch();

closegraph();

}

**OUTPUT**

****

****

**6.Write a Program for translation of an object in graphics.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

class transform //create class

{

public:

void translation(int[],int,int,int);

};

void main()

{

clrscr();

int gd=DETECT,gm;

int fig[20],edges,dx,dy,i; //declaration of variables

transform t; //create object

cout<<"Enter number of edges"<<endl;

cin>>edges; //read no. of edges

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

{

cout<<"Enter edges((x"<<i<<",y"<<i<<"))"<<endl;

cin>>fig[2\*i]>>fig[2\*i+1]; //read edges in fig[]

}

fig[2\*i]=fig[0];

fig[2\*i+1]=fig[1];

edges+=1;

cout<<"Enter dx"<<endl;

cin>>dx; //read value of x

cout<<"Enter dy"<<endl;

cin>>dy; //read value of y

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

cleardevice();

drawpoly(edges,fig); //calling drawpoly()

getch();

t.translation(fig,edges,dx,dy); //call class function

setcolor(BLUE);

drawpoly(edges,fig); //calling drawpoly()

getch();

}

void transform::translation(int fig[],int edges,int dx,int dy) //function definition

{

int i;

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

{

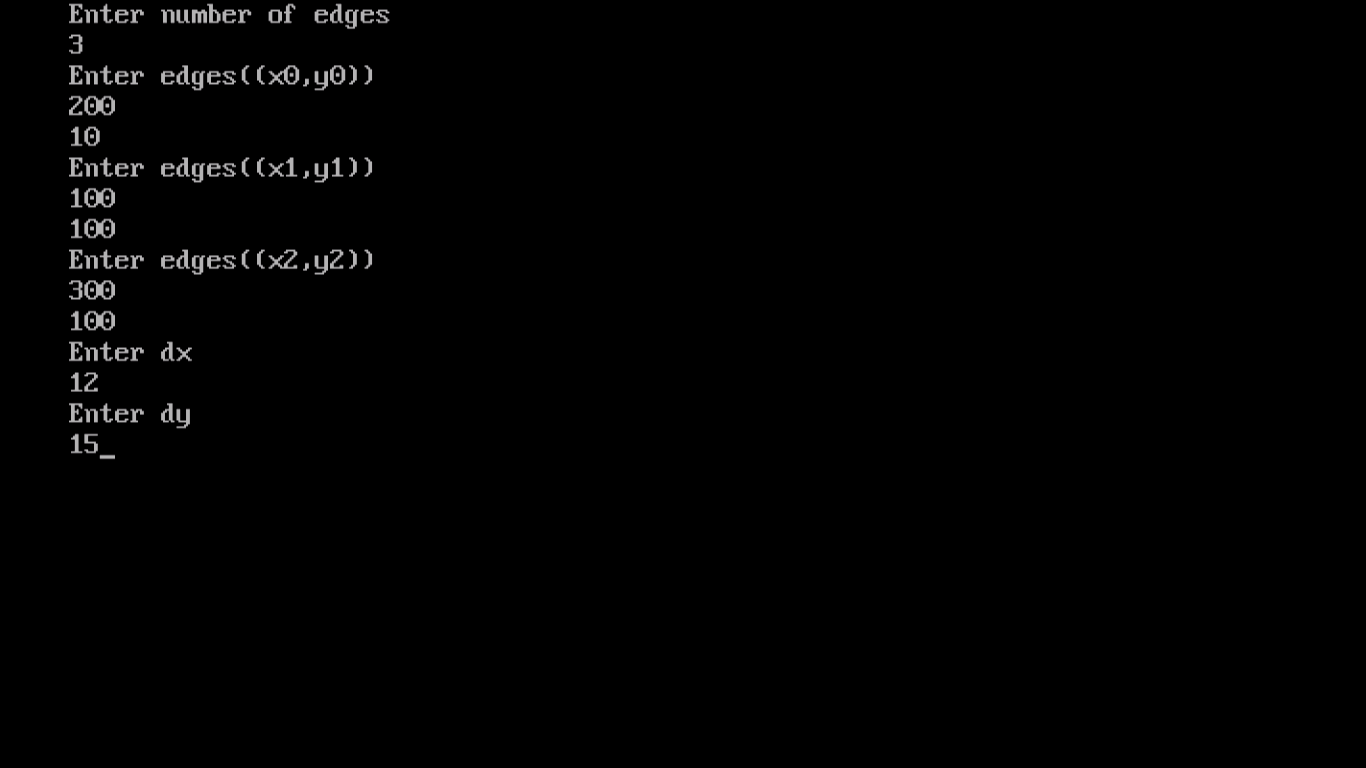
fig[2\*i]+=dx;

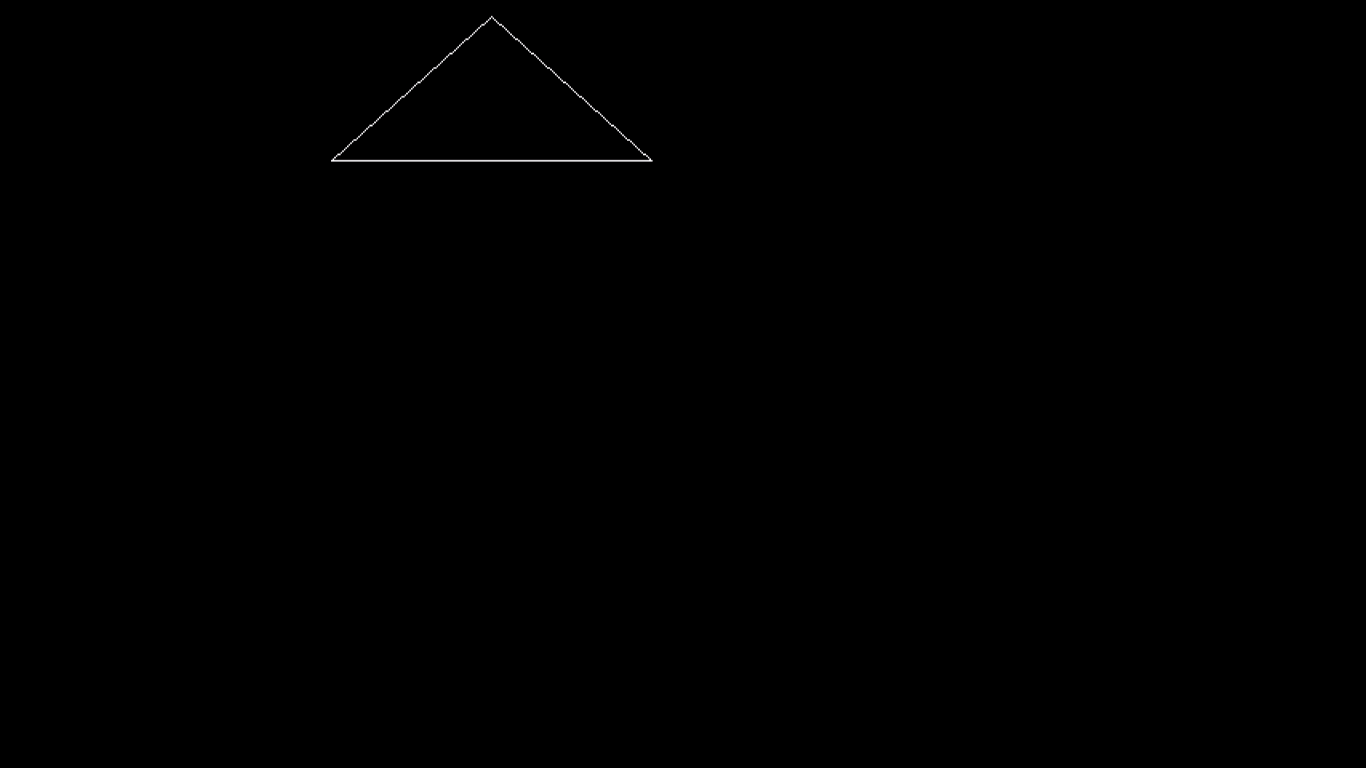
fig[2\*i+1]+=dy;

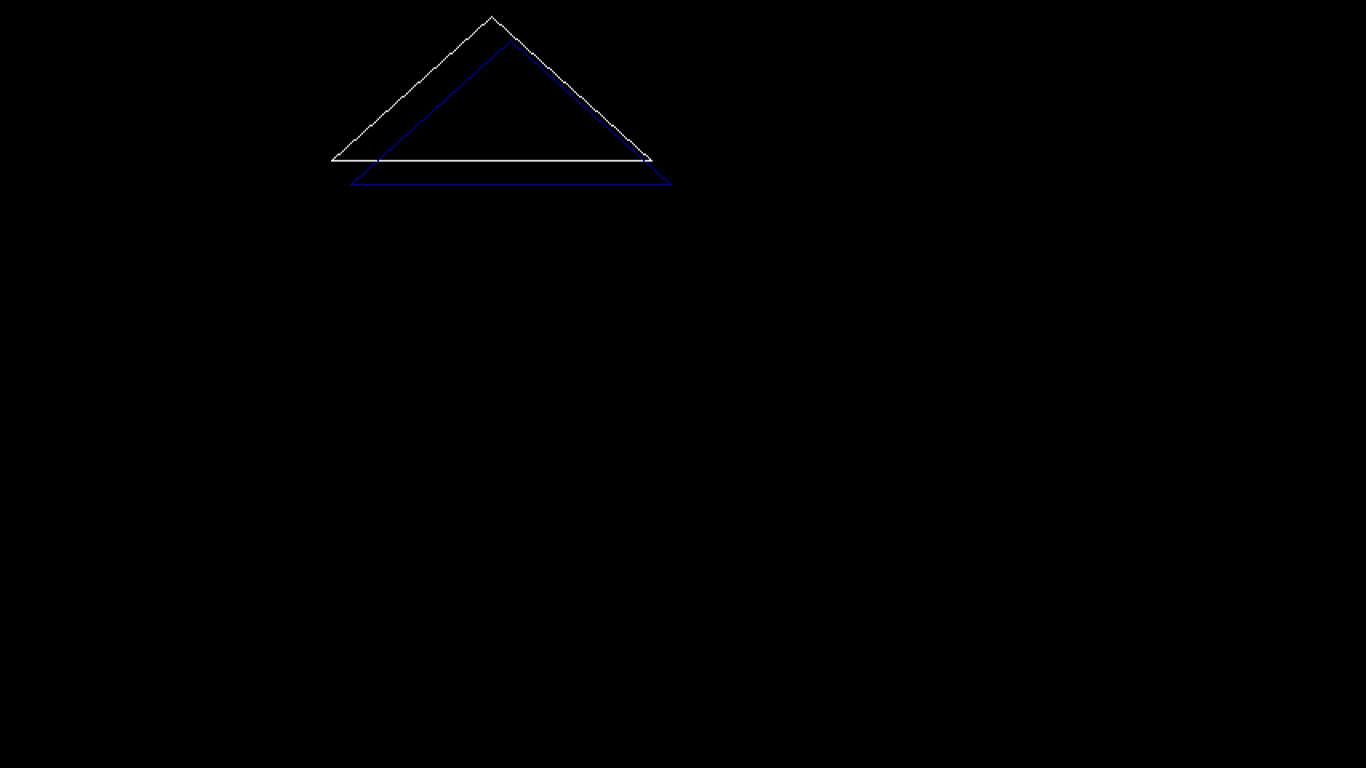
}

}

**OUTPUT**

****

****

****

**7.Write a Program for rotation of an object according to angle in graphics.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

class transform //create class

{

public:

void rotate(int[],int,double,int,int);

};

void main()

{

clrscr();

int gd=DETECT,gm;

int i,edges,cx=0,cy=0;

double angle;

int fig[20]; //declaration of variables

transform t; //create object

cout<<"Enter number of edges"<<endl;

cin>>edges; //read no. of edges

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

{

cout<<"Enter edges((x"<<i<<",y"<<i<<"))"<<endl;

cin>>fig[2\*i]>>fig[2\*i+1]; //read edges in fig[]

}

fig[2\*i]=fig[0];

fig[2\*i+1]=fig[1];

edges+=1;

cout<<"Enter the angle of rotation"<<endl;

cin>>angle; //read angle

cout<<"Enter center of rotation cx"<<endl;

cin>>cx; //read cx

cout<<"Enter center of rotation cy"<<endl;

cin>>cy; //read cy

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

cleardevice();

setbkcolor(WHITE); //set background color

setcolor(GREEN); //set color

setlinestyle(SOLID\_LINE,0,3); //set linestyle

drawpoly(edges,fig); //calling drawpoly()

getch();

t.rotate(fig,edges,angle,cx,cy); //call class fucntion

getch();

}

void transform::rotate(int fig[],int edges,double angle,int cx,int cy) //function definition

{

double x,y,cosa,sina;

int i;

cout<<"angle="<<angle<<endl;

angle=(angle\*3.14/180);

cout<<angle<<endl;

cosa=cos(angle);

sina=sin(angle);

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

{

fig[2\*i]=(cx+((fig[2\*i]-cx)\*cosa-(fig[2\*i+1]- cy)\*sina));

fig[2\*i+1]=(cx+((fig[2\*i]-cx)\*sina-(fig[2\*i+1]- cy)\*cosa));

cout<<fig[2\*i]<<endl;

cout<<fig[2\*i+1]<<endl;

}

fig[2\*i]=fig[0];

fig[2\*i+1]=fig[1];

edges+=1;

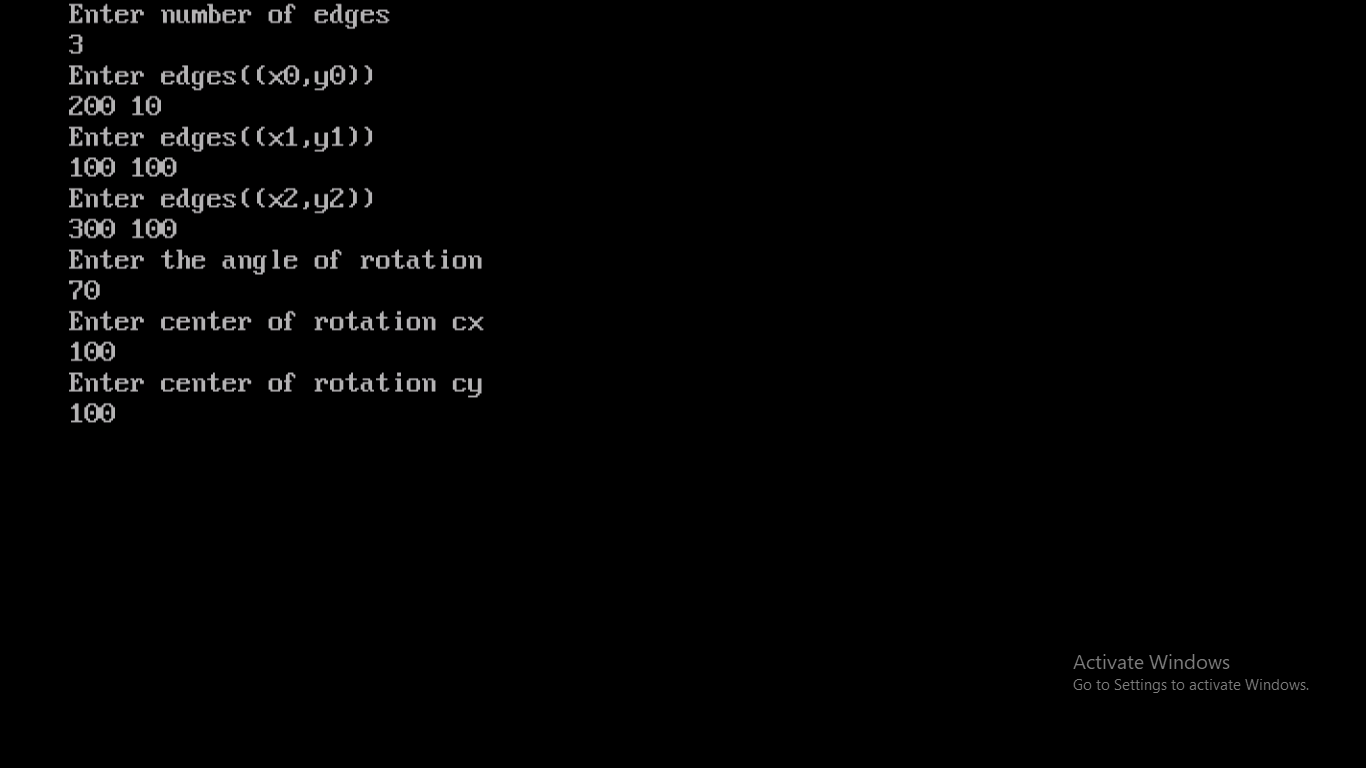
setcolor(RED); //set the color

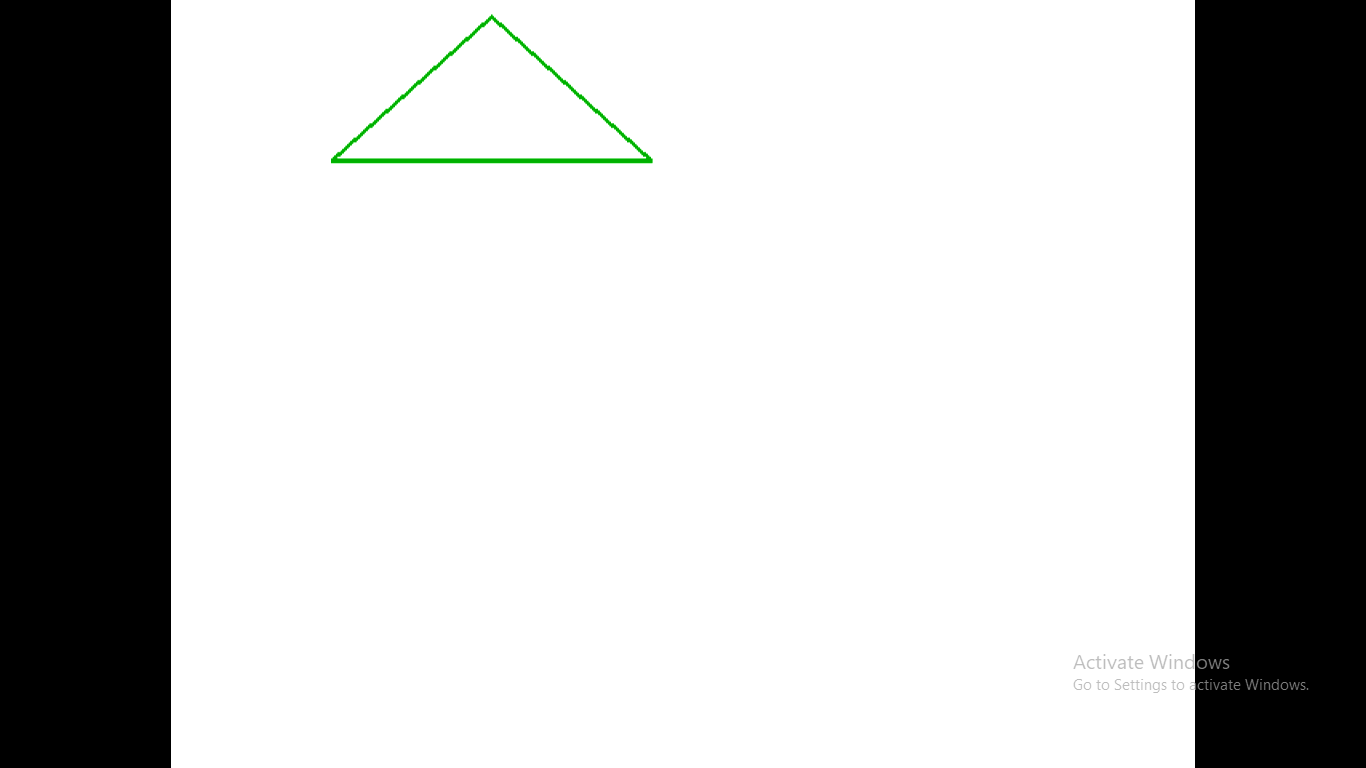
setlinestyle(SOLID\_LINE,0,3);

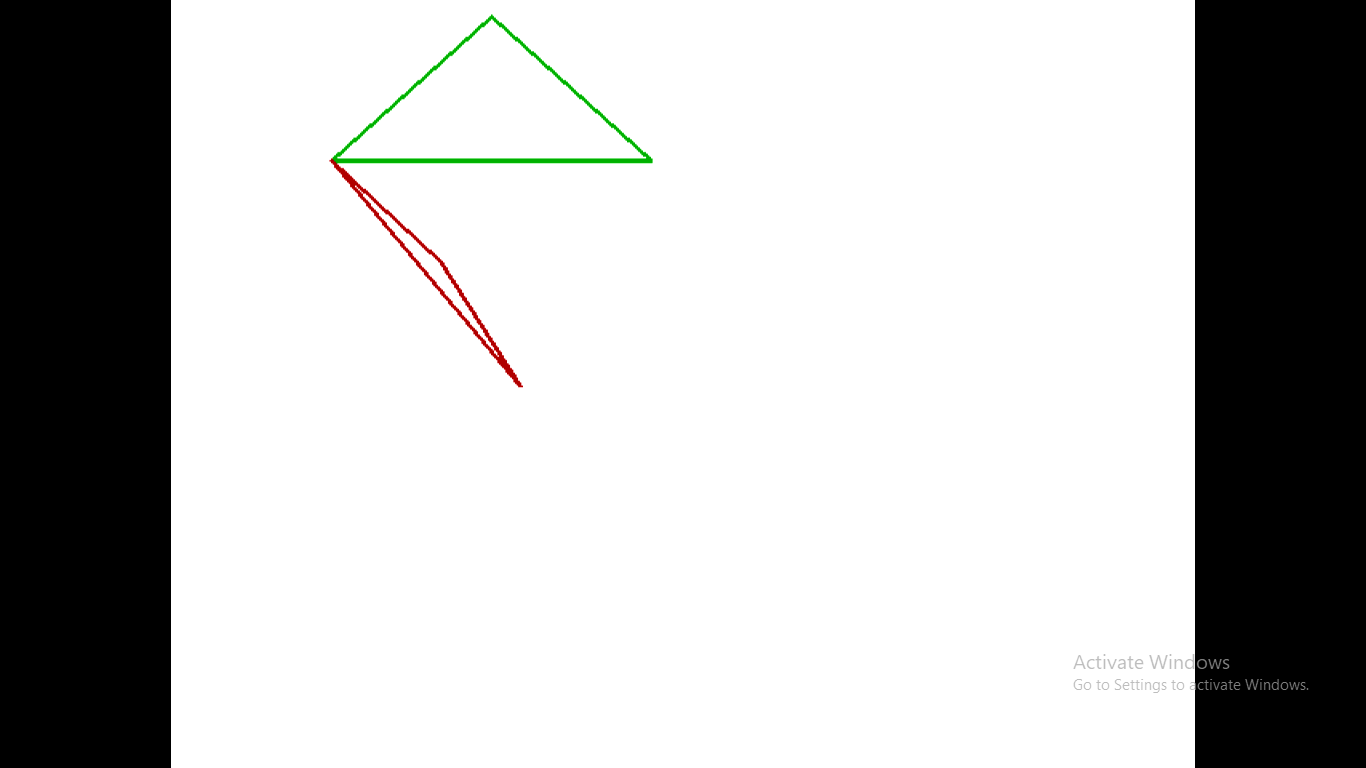
drawpoly(edges,fig); //calling drawpoly()

}

**OUTPUT**

****

****

****

**8.Write a Program of shearing of an object according to x-axis,y-axis in graphics.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<stdlib.h>

int i,maxx,maxy,x1,y1,x2,y2,x3,y3,x4,y4,gap=50,option;

float shx=0.0,shy=0.0;

char str[5]; //global variable declaration

class shear //create class

{

public:

void shearing(); //function declaration

};

void shear::shearing() //function definition

{

clearviewport();

maxx=getmaxx();

maxy=getmaxy();

line(3,maxy-1,maxx-5,maxy-1);

line(5,5,5,maxy-3);

for(i=0;i<maxx-5;i=i+gap)

{

outtextxy(i+3,maxy-7,"|");

itoa(i,str,10);

outtextxy(i,maxy-10,str);

}

for(i=maxy;i>0;i=i-gap)

{

outtextxy(3,i,"-");

itoa(maxy-i,str,10);

outtextxy(9,i,str);

}

setcolor(50);

line(x1,maxy-y1,x2,maxy-y2);

line(x3,maxy-y3,x4,maxy-y4);

line(x1,maxy-y1,x3,maxy-y3);

line(x2,maxy-y2,x4,maxy-y4);

outtextxy(10,10,"hit any key");

getch();

setcolor(0);

line(x1,maxy-y1,x2,maxy-y2);

line(x3,maxy-y3,x4,maxy-y4);

line(x1,maxy-y1,x3,maxy-y3);

line(x2,maxy-y2,x4,maxy-y4);

setcolor(58);

if(option==1)

{

line(x1+shx\*y1,maxy-y1,x2+shx\*y2,maxy-y2);

line(x3,maxy-y3,x4,maxy-y4);

line(x1+shx\*y1,maxy-y1,x3,maxy-y3);

line(x2+shx\*y2,maxy-y2,x4,maxy-y4);

}

else

{

line(x1,maxy-y1,x2,maxy-(y2+shy\*x2));

line(x3,maxy-y3,x4,maxy-(y4+shy\*x4));

line(x1,maxy-y1,x3,maxy-y3);

line(x2,maxy-(y2+shy\*x2),x4,maxy-(y4+shy\*x4));

}

}

void main()

{

clrscr();

int gd=DETECT,gm,xref,yref;

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

cout<<"Enter endpoints of top corner"<<endl;

cin>>x1>>y1>>x2>>y2; //read top corner values

cout<<"Enter endpoints of bottom corner"<<endl;

cin>>x3>>y3>>x4>>y4; //read bottom corner values

cout<<"Enter 1 for x axis shear"<<endl;

cout<<"Enter 2 for y axis shear"<<endl;

cin>>option;

if(option==1)

{

cout<<"Enter the value of x axis shear"<<endl;

cin>>shx;

}

else

{

cout<<"Enter the value of y axis shear"<<endl;

cin>>shy;

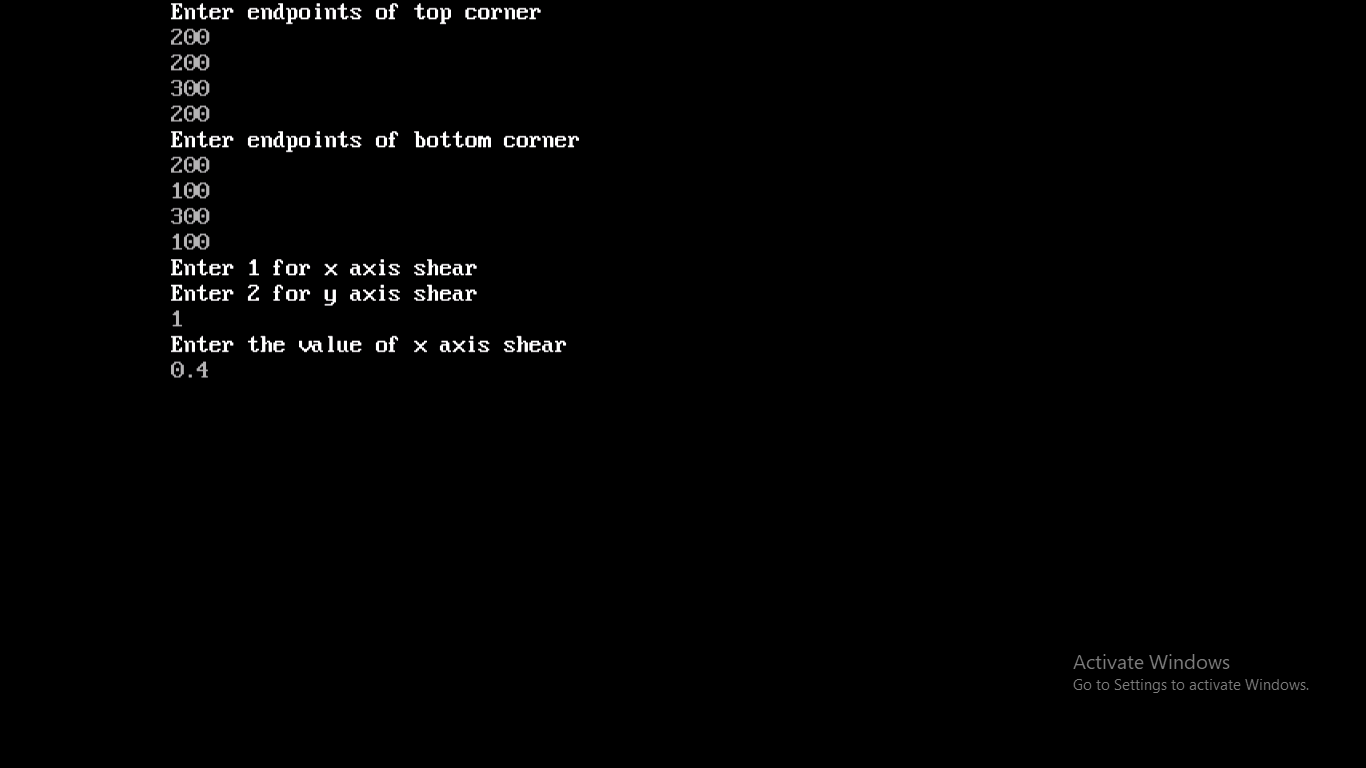
}

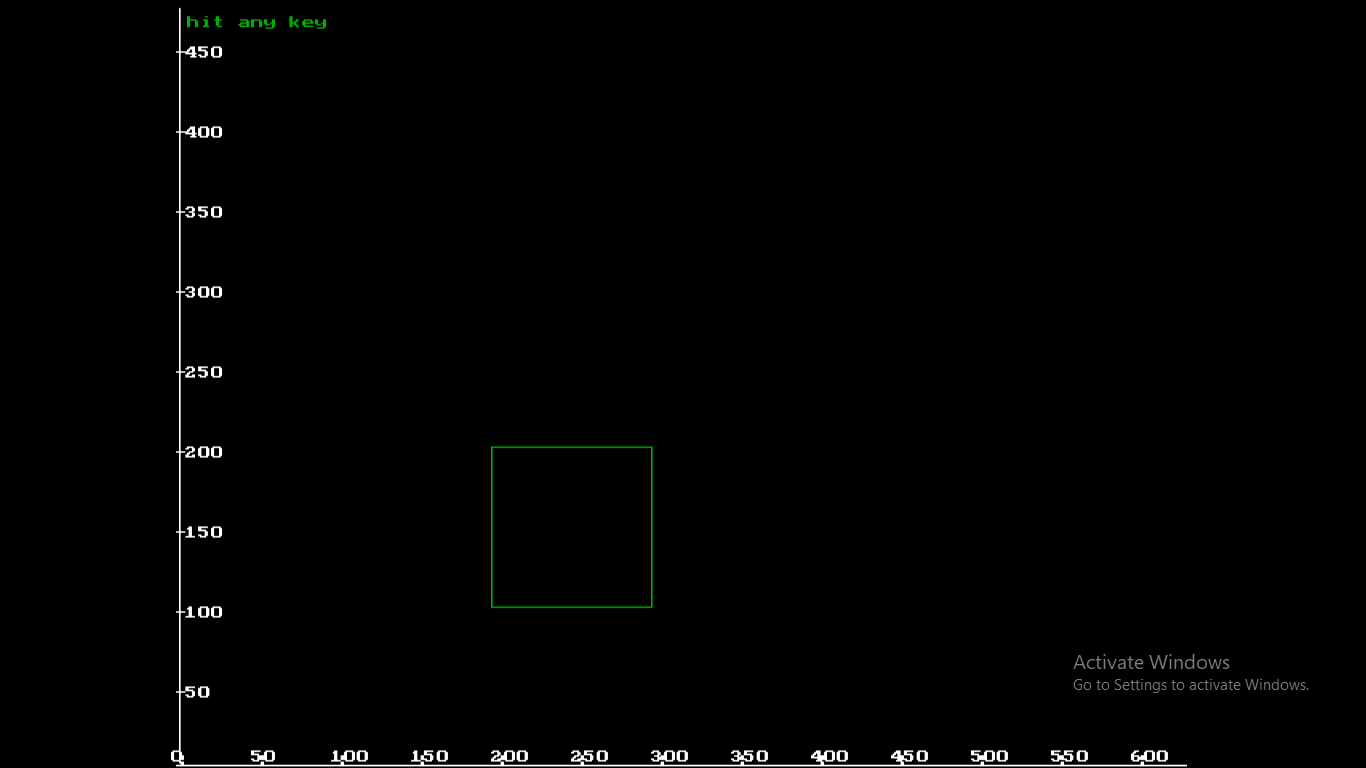
shear s; //create object

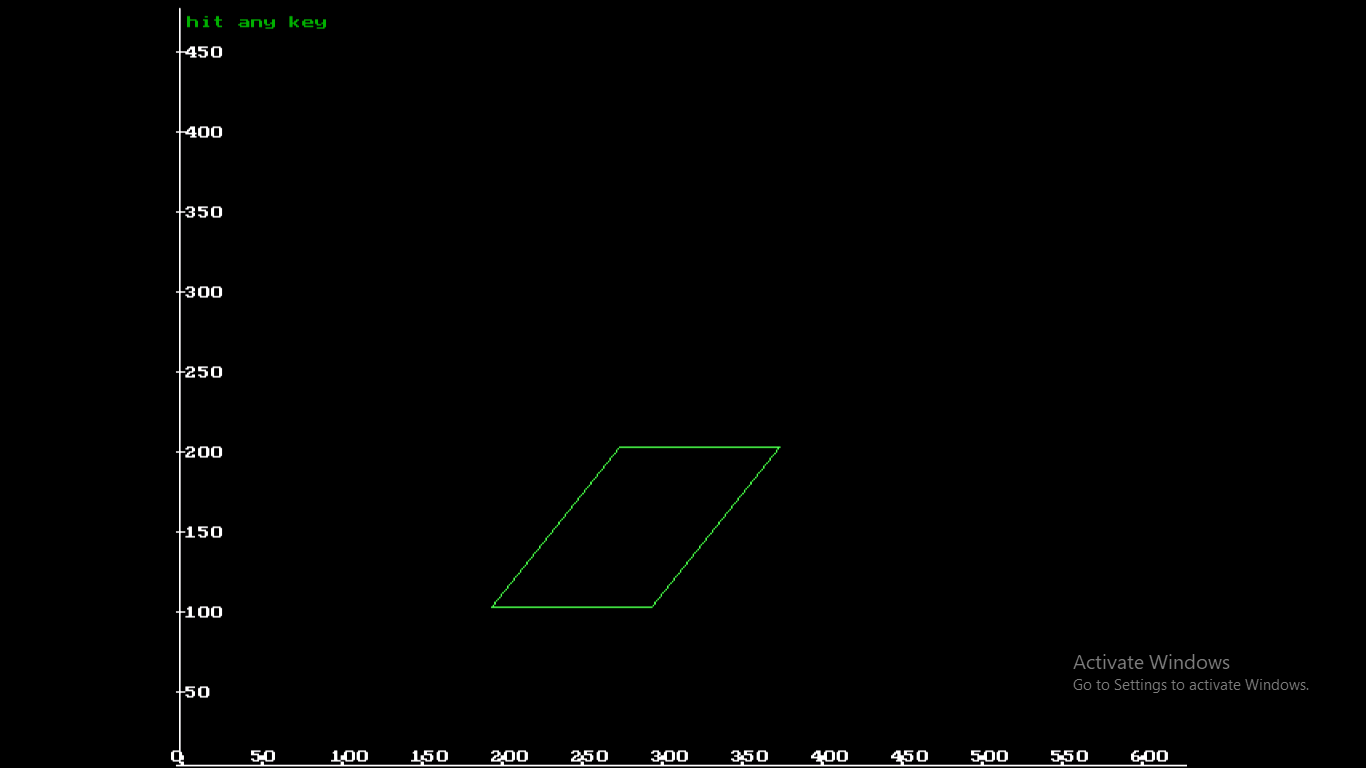
s.shearing(); //call function with object

getch();}

**OUTPUT**







**9.Write a Program for reflection of an object according to x-axis,y-axis,x=y in graphics.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

int ch,i,edges,fig[20],fig1[20];

class transform //create class

{

public:

void reflection(); //function declaration

};

void transform::reflection() //function definition

{

cout<<"Enter your choice for reflection among different axes"<<endl;

cout<<"Enter 1 along x-axis"<<endl;

cout<<"Enter 2 along y-axis"<<endl;

cout<<"Enter 3 along x=y axis"<<endl;

cin>>ch; //enter choice

switch(ch) //switch case

{

case 1:

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

{

fig1[2\*i]=fig[2\*i];

fig1[2\*i+1]=(240-fig[2\*i+1])+240;

}

break;

case 2:

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

{

fig1[2\*i]=320-(fig[2\*i]-320);

fig1[2\*i+1]=fig[2\*i+1];

}

break;

case 3:

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

{

fig1[2\*i]=320-(fig[2\*i]-320);

fig1[2\*i+1]=240-(fig[2\*i+1]-240);

}

break;

}

fig1[2\*i]=fig1[0];

fig1[2\*i+1]=fig1[1];

edges+=1;

getch();

setcolor(YELLOW);

drawpoly(edges,fig1); //calling drawpoly()

}

void main()

{

clrscr();

int gd=DETECT,gm;

int dx,dy;

int xmax,ymax;

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

line(0,240,639,240);

line(320,0,320,479);

cout<<"Enter number of edges"<<endl;

cin>>edges; //read number of edges

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

{

cout<<"Enter edges((x"<<i<<",y"<<i<<"))"<<endl;

cin>>fig[2\*i]>>fig[2\*i+1]; //read edges in fig[]

}

fig[2\*i]=fig[0];

fig[2\*i+1]=fig[1];

edges+=1;

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

{

fig[2\*i]=fig[2\*i]+320;

}

drawpoly(edges,fig);

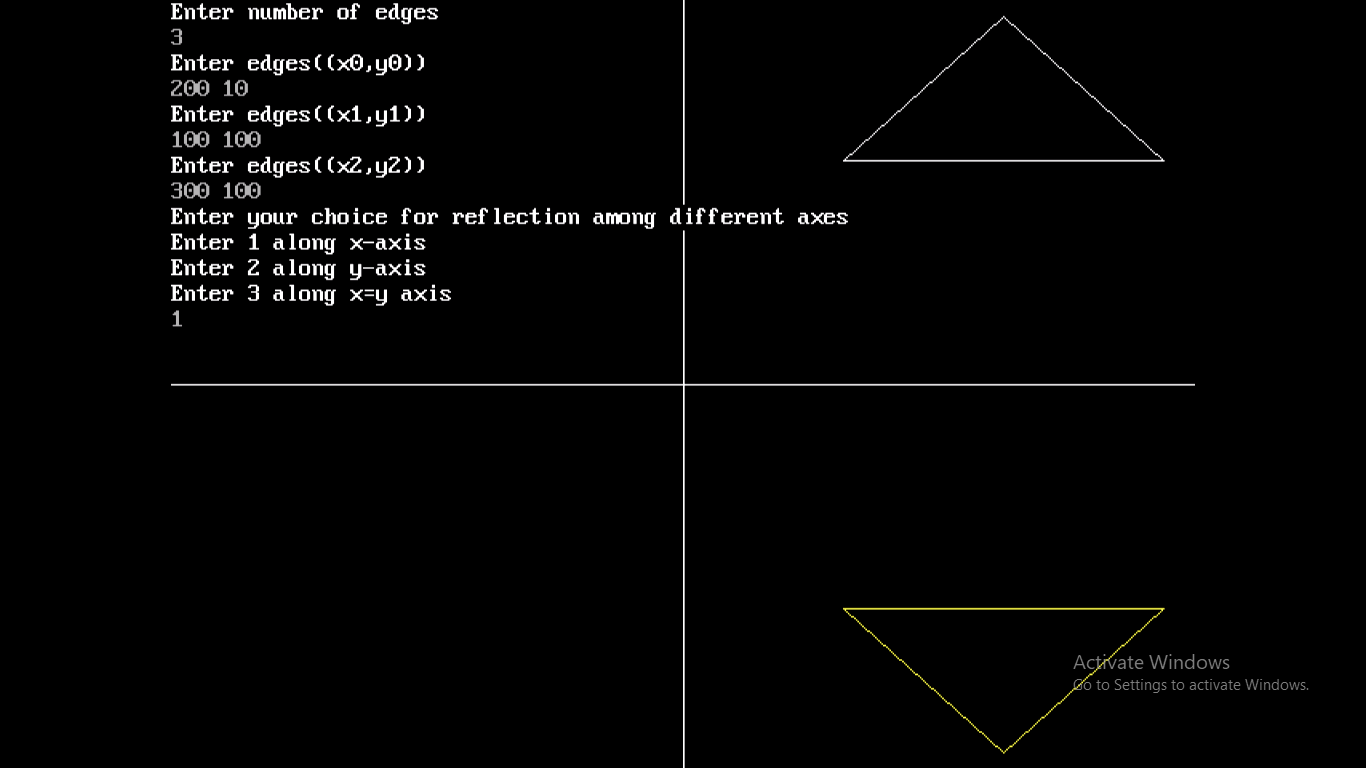
transform t; //create object

t.reflection(); //call function with object

getch();

}

**OUTPUT**



**10.Write a Program of ellipse using midpoint method in graphics.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<dos.h>

class Mid //create class

{

public:

void ellipse(); //function declaration

};

void Mid::ellipse() //function definition

{

float dx,dy,d1,d2,x,y,h,k,rx,ry;

cout<<"Enter the center point of ellipse(h,k)"<<endl;

cin>>h>>k;

cout<<"Enter the max and min radius of ellipse"<<endl;

cin>>rx>>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;

while(dx<dy)

{

//plot the points

putpixel(x+h,y+k,RED);

putpixel(-x+h,y+k,RED);

putpixel(x+h,-y+k,RED);

putpixel(-x+h,-y+k,RED);

if(d1<0)

{

x=x+1;

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);

}

}

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

while(y>=0)

{

//plot the points

putpixel(x+h,y+k,RED);

putpixel(-x+h,y+k,RED);

putpixel(x+h,-y+k,RED);

putpixel(-x+h,-y+k,RED);

if(d2>0)

{

y=y-1;

dy=dy-(2\*rx\*rx);

d2=d2+(rx\*rx)-dy;

}

else

{

y=y-1;

x=x+1;

dx=dx+(2\*ry\*ry);

dy=dy-(2\*rx\*rx);

d2=d2+dx-dy+(rx\*rx);

}

}

}

void main()

{

int gd=DETECT,gm;

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

Mid e; //object create

e.ellipse(); //function call

getch();

}

**OUTPUT**

