PRACTICAL-1(A)

- **putpixel():** The header file graphics.h contains **putpixel()** function which plots a pixel at location (x, y) of specified color.
  Syntax : `void putpixel(int x, int y, int color);`

where, (x, y) is the location at which pixel is to be put , and color specifies the color of the pixel.

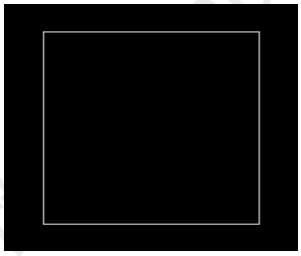
- **Draw Rectangle:** **rectangle()** is used to draw a rectangle. Coordinates of left top and right bottom corner are required to draw the rectangle. left specifies the X-coordinate of top left corner, top specifies the Y-coordinate of top left corner, right specifies the X-coordinate of right bottom corner, bottom specifies the Y-coordinate of right bottom corner.
  Syntax :
  `rectangle(int left, int top, int right, int bottom);`

Examples1:
Input : left = 150, top = 250, right = 450, bottom = 350;

Output :

Examples2:
Input : left = 150, top = 150, right = 450, bottom = 450;

Output :



- **Draw a line**

Examples:
For line 1, Input : x1 = 150, y1 = 150, x2 = 450, y2 = 150

For line 2, Input : x1 = 150, y1 = 200, x2 = 450, y2 = 200

For line 2, Input : x1 = 150, y1 = 250, x2 = 450, y2 = 250

Output :

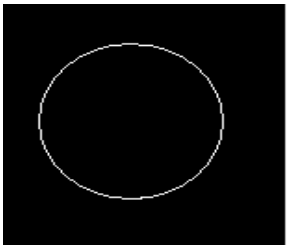
Draw circle : The header file graphics.h contains **circle()** function which draws a circle with center at (x, y) and given radius.

Syntax : `circle(x, y, radius);`

where,
(x, y) is center of the circle.
'radius' is the Radius of the circle.

Examples : `Input : x = 250, y = 200, radius = 50`

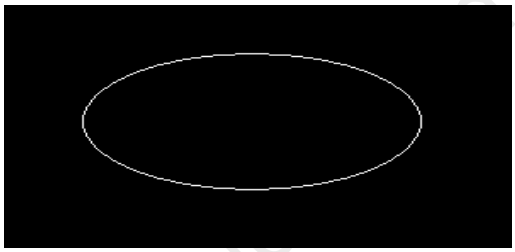
Output :



- **Draw ellipse:** Program to draw ellipse in C using graphics.h header file**.** graphics.h library is used to include and facilitate graphical operations in program. C graphics using graphics.h functions can be used to draw different shapes, display text in different fonts, change colors and many more. Using functions of graphics.h you can make graphics programs, animations, projects and games. You can draw circles, lines, rectangles, bars and many other geometrical figures. You can change their colors using the available functions and fill them.

Examples:

```
Input : x=250, y=200, start_angle = 0,
        end_angle = 360, x_rad = 100, y_rad = 50
Output :
```



- **getmaxx():** The header file graphics.h contains **getmaxx()** function which returns the maximum X coordinate for current graphics mode and driver.
  Syntax : `int getmaxx();`

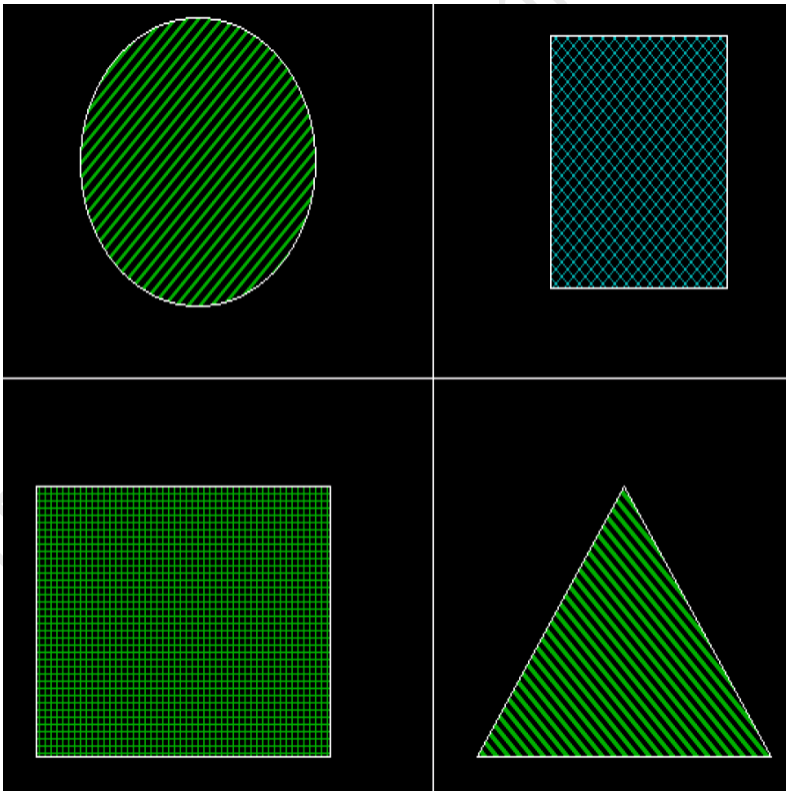
- **getmaxy():** The header file graphics.h contains **getmaxy()** function which returns the maximum Y coordinate for current graphics mode and driver.
  Syntax : `int getmaxy();`

PRACTICAL-2(A.1)

Divide screen in to four regions, draw circle, rectangle and triangle square in each region and fill the color.

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
clrscr();
initgraph(&gd,&gm,"c:\\tc\\bgi");
line(1,240,640,240);
line(320,1,320,480);
circle(160,120,80);
setfillstyle(4,2);
floodfill(160,120,15);
rectangle(400,50,520,190);
setfillstyle(8,3);
floodfill(450,80,15);
rectangle(50,300,250,450);
setfillstyle(1,2);
floodfill(80,320,15);
line(450,300,350,450);
line(450,300,550,450);
line(350,450,550,450);
setfillstyle(1,2);
floodfill(450,400,15);
getch();
}
```

OUTPUT

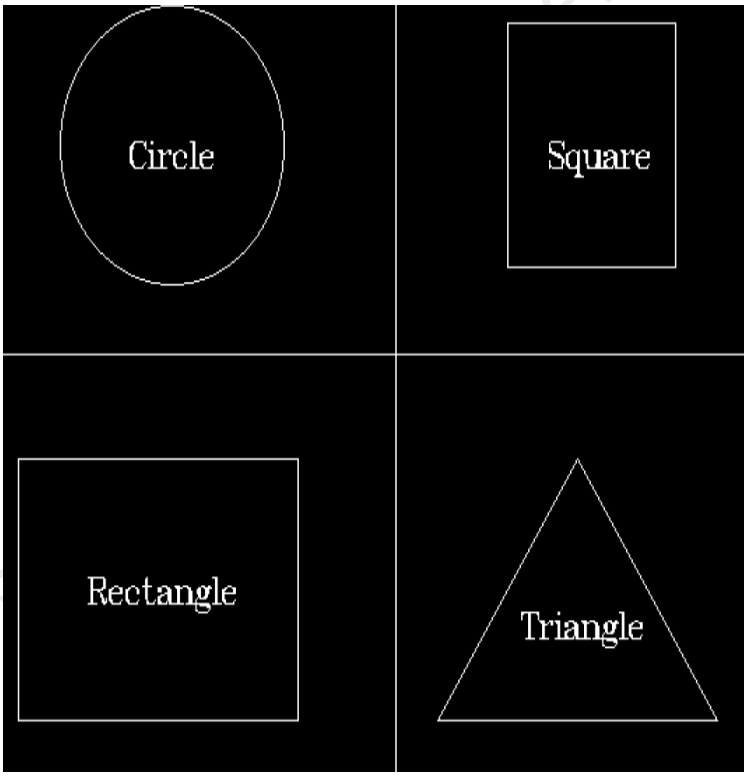


PRACTICAL-2(A)

Divide screen in to four regions, draw circle, rectangle and triangle square in each region with appropriate message.

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm,i;
clrscr();
initgraph(&gd,&gm,"c:\\tc\\bgi");
for(i=2;i<10;i++)
{
line(1,240,640,240);
line(320,1,320,480);
circle(160,120,80);
//setfillstyle(4,2);
//floodfill(160,120,15);
//setcolor(RED);
settextstyle(1,0,3);
outtextxy(130,110,"Circle");
rectangle(400,50,520,190);
//setfillstyle(8,3);
//floodfill(450,80,15);
outtextxy(430,110,"Square");
rectangle(50,300,250,450);
//setfillstyle(7,2);
//floodfill(80,320,15);
outtextxy(100,360,"Rectangle");
line(450,300,350,450);
line(450,300,550,450);
line(350,450,550,450);
//setfillstyle(5,2);
//floodfill(450,400,15);
outtextxy(410,380,"Triangle");
}
```

OUTPUT

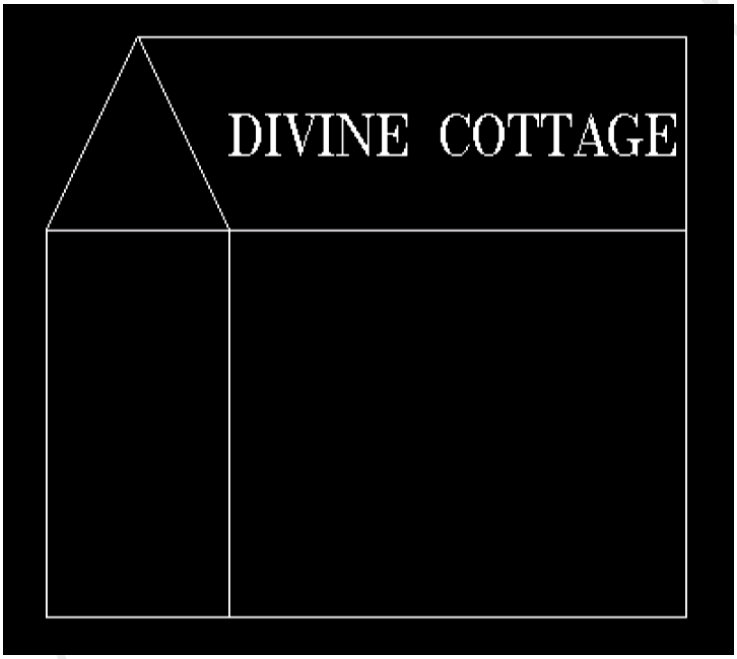


PRACTICAL-2(B)

Draw a Simple HUT on the screen

```
#include<iostream.h>
 #include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm,xcen,ycen;
clrscr();
initgraph(&gd,&gm,"c:\\tc\\bgi");
settextstyle(1,0,4);
outtextxy(150,130,”DIVINE COTTAGE”);
line(100,100,400,100);
line(100,100,50,200);
line(50,200,400,200);
line(400,400,400,200);
line(400,100,400,200);
line(150,200,150,400);
line(100,100,150,200);
rectangle(50,200,400,400);
getch();
}
```

OUTPUT



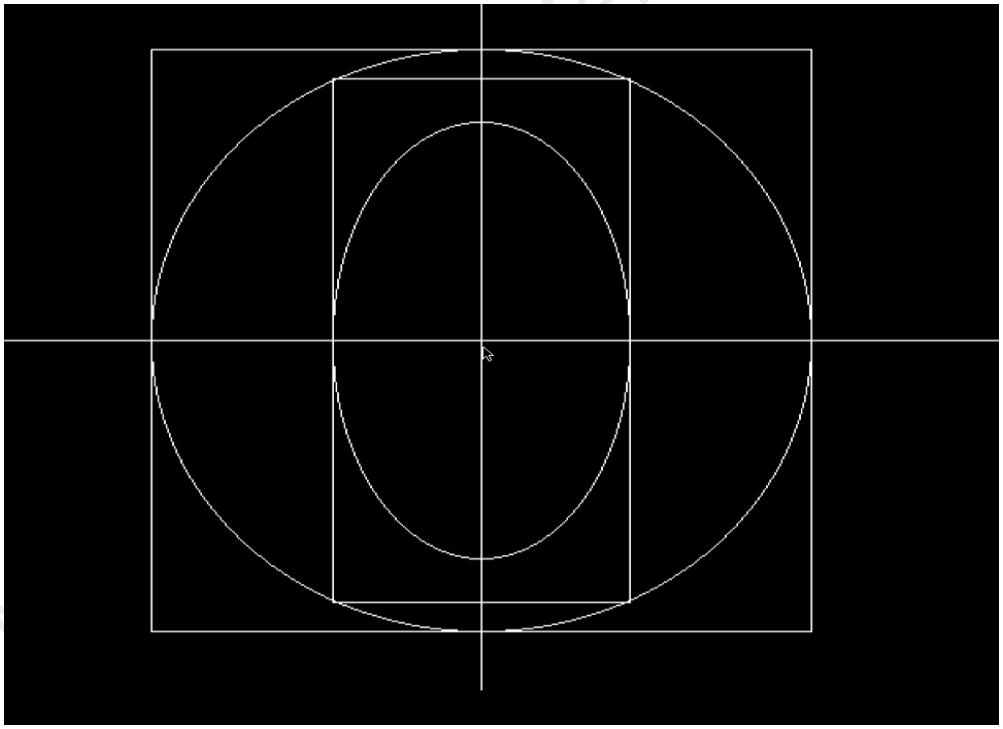
PRACTICAL-3(a)

Draw the following shapes in the center of the screen:

1. Line 2. Square 3.Circle 4.Reactangle 5.Ellipse.

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm,xcen,ycen;
initgraph(&gd,&gm,"c:\\tc\\bgi");
xcen=getmaxx()/2;
ycen=getmaxy()/2;
line(xcen,0,xcen,getmaxx());
line(0,ycen,getmaxx(),ycen);
circle(xcen,ycen,200);
rectangle(xcen-200,ycen-200,xcen+200,ycen+200); //square
rectangle(xcen-90,ycen-180,xcen+90,ycen+180);
ellipse(xcen,ycen,0,360,90,150);
getch();
}
```

OUTPUT

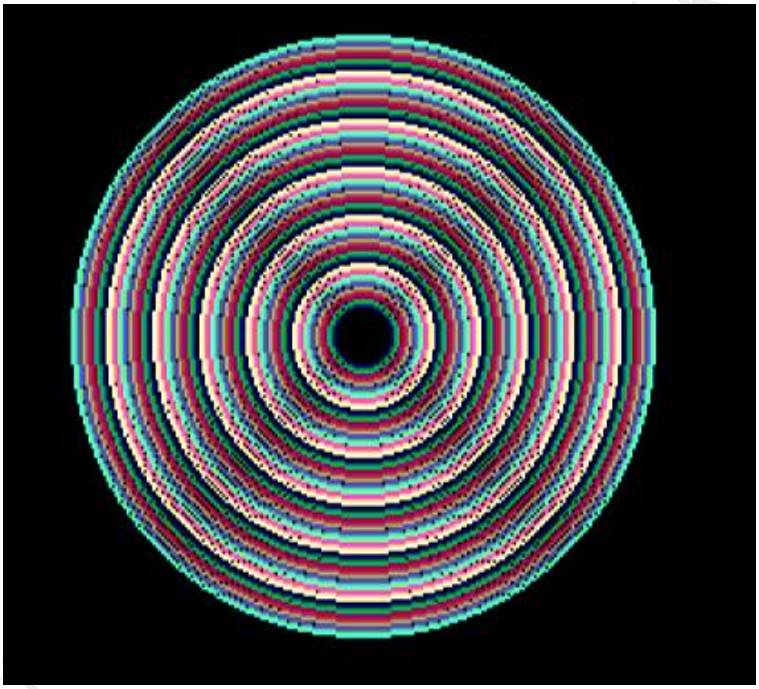


PRACTICAL-3(b)

Draw the following Concentric Circles in the center of the screen

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
int gd=DETECT,gm,xcen,ycen,i,color=1;
initgraph(&gd,&gm,"c:\\tc\\bgi");
xcen=getmaxx()/2;
ycen=getmaxy()/2;
for(i=10;i<=100;i+=1)
{
setcolor(color++);
circle(xcen,ycen,i);
delay(20);
}
getch();
}
```

OUTPUT

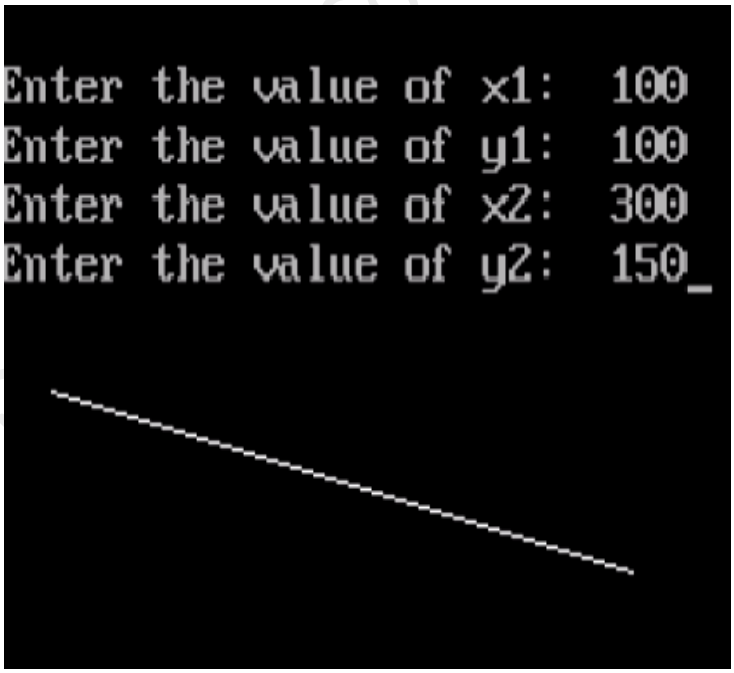


PRACTICAL-4(A)

DDA LINE drawing algorithm.

```
#include<iostream.h>
#include<conio.h>
#include<graphic.h>
#include<dos.h>
#include<maths.h>

void main()
{
        float x,y,x1,y1,x2,y2,dx,dy,length;
        int i,gd,gm;
        clrscr();
        cout<<"\nenter the value of x1 :\t";
        cin>>x1;
        cout<<"\nenter the value of y1 :\t";
        cin>>y1;
        cout<<"\nenter the value of x2 :\t";
        cin>>x2;
        cout<<"\nenter the value of y2 :\t";
        cin>>y2;
        detectgraph(&gd,&gm);
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        dx=abs(x2-x1);
        dy=abs(y2-y1);
        if(dx>=dy)
        {
                length=dx;
        }
        dx=(x2-x1)/length;
        dy=(y2-y1)/length;
        x=x1+0.5;
        y=y1+0.5;
        i=1;

        while(i<=length)
        {
                putpixel(x,y,6);
                x=x+dx;
                y=y+dy;
                i=i+1;
                delay(100);
        }
        getch();
    }
```

OUTPUT :

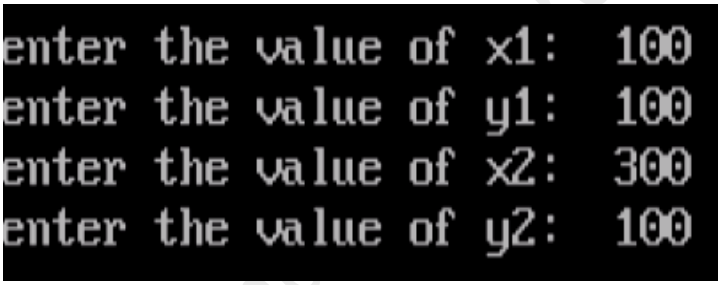


PRACTICAL- 4(B)

Bresenham's Line drawing algorithm

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
void main()
{
        float x,y,x1,y1,x2,y2,dx,dy,e;
        int i,gd,gm;
        clrscr();
        cout<<"enter the value of x1:\t";
        cin>>x1;
        cout<<"enter the value of y1:\t";
        cin>>y1;
        cout<<"enter the value of x2:\t";
        cin>>x2;
        cout<<"enter the value of y2:\t";
        cin>>y2;
        detectgraph(&gd,&gm);
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        dx=abs(x2-x1);
        dy=abs(y2-y1);
        x=x1;
        y=y1;
        e=2*(dy-dx);
        i=1;
        do
        {
                putpixel(x,y,15);
                        while(e>0)
                        {
                                y=y+1;
                                e=e-2*dx;
                        }
                        x=x+1;
                        e=e-2*dy;
                        i=i+1;
                        delay(50);
        }while(i<=dx);
        getch();
}
```

OUTPUT :

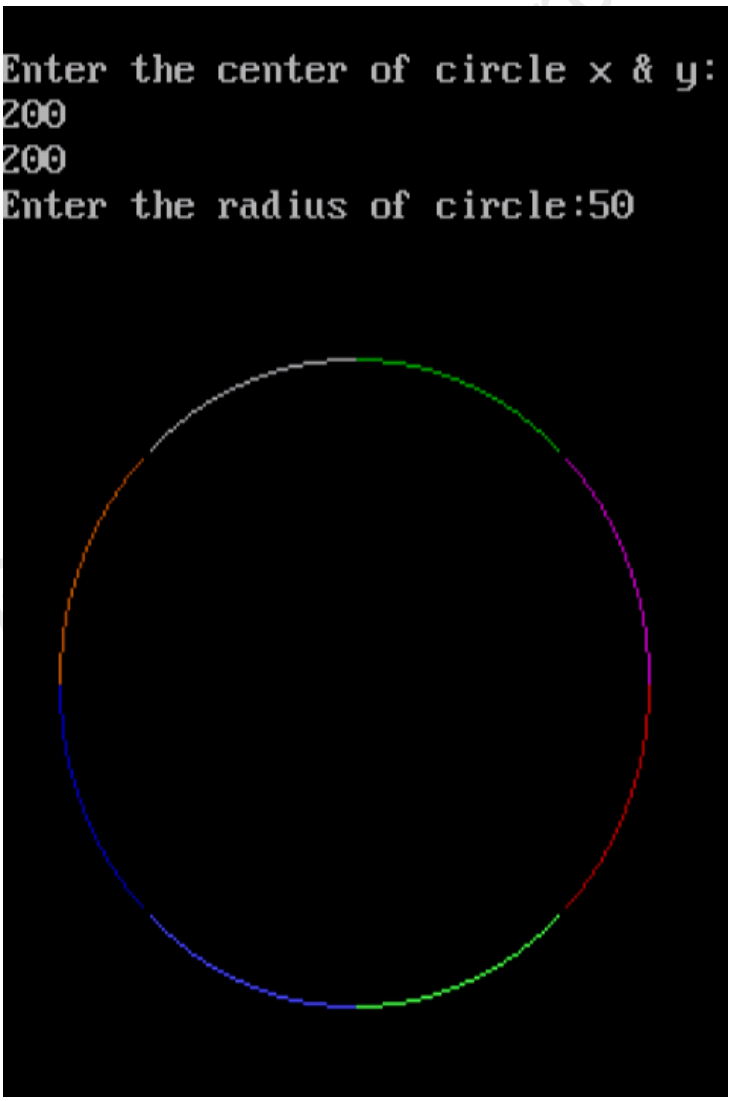


PRACTICAL-5(A)

Mid-Point Circle Drawing Algorithm.

```
#include<iostream.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>
void plotpoints(int,int,int,int);
void main()
{
        int x,y,x1,y1,p,r;
        int gd,gm;
        clrscr();
        cout<<"Enter the center of circle x & y:";
        cin>>x>>y;
        cout<<"Enter the radius of circle:";
        cin>>r;
        detectgraph(&gd,&gm);
        initgraph(&gd,&gm,"c:\\tc\\bgi");
        x1=0;
        y1=r;
        p=1-r;
        while(x1<=y1)
        {
                plotpoints(x,y,x1,y1);
                if(p<0)
                {
                        p=p+2*x1+3;
                }
                else
                {
                        p=p+2*(x1-y1)+5;
                        y1--;
                }
                x1++;
        }
        getch();
}
void plotpoints(int a,int b,int c,int d)
{
        putpixel(a+c,b+d,10);
        putpixel(a-c,b+d,9);
        putpixel(a+c,b-d,2);
        putpixel(a-c,b-d,7);
        putpixel(a+d,b+c,RED);
        putpixel(a-d,b+c,BLUE);
        putpixel(a+d,b-c,5);
        putpixel(a-d,b-c,6);
        delay(100);
}
```

OUTPUT:



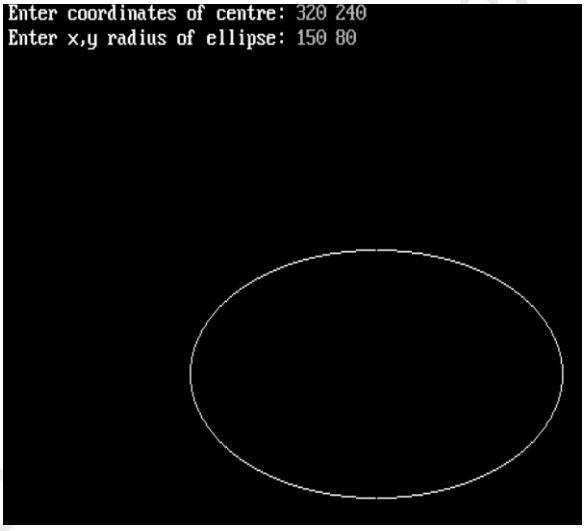
PRACTICAL-5(b)

//w.a.p for Mid-Point ellipse drawing algorithm.

```
#include<graphics.h>
#include<stdlib.h>
#include<iostream.h>
#include<conio.h>
void main ()
{
  clrscr();
  int gd = DETECT, gm;
  int xc,yc,x,y;float p;
  long rx,ry;
  initgraph(&gd, &gm, "C:\\TC\\BGI");
  cout<<"Enter coordinates of centre : ";
  cin>>xc>>yc;
  cout<<"Enter x,y radius of ellipse: ";
  cin>>rx>>ry;
  //Region 1
  p=ry*ry-rx*rx*ry+rx*rx/4;
  x=0;y=ry;
  while(2.0*ry*ry*x <= 2.0*rx*rx*y)
  {
        if(p < 0)
        {
              x++;
              p = p+2*ry*ry*x+ry*ry;
        }
        else
        {
              x++;y--;
              p = p+2*ry*ry*x-2*rx*rx*y-ry*ry;
        }
        putpixel(xc+x,yc+y,RED);
        putpixel(xc+x,yc-y,RED);
        putpixel(xc-x,yc+y,RED);
        putpixel(xc-x,yc-y,RED);
  }
```

```
//Region 2
  p=ry*ry*(x+0.5)*(x+0.5)+rx*rx*(y-1)*(y-1)-rx*rx*ry*ry;
  while(y > 0)
  {
        if(p <= 0)
        {
              x++;y--;
              p = p+2*ry*ry*x-2*rx*rx*y+rx*rx;
        }
        else
        {
              y--;
              p = p-2*rx*rx*y+rx*rx;
        }
        putpixel(xc+x,yc+y,RED);
        putpixel(xc+x,yc-y,RED);
        putpixel(xc-x,yc+y,RED);
        putpixel(xc-x,yc-y,RED);
  }
  getch();
  closegraph();
}
```

OUTPUT

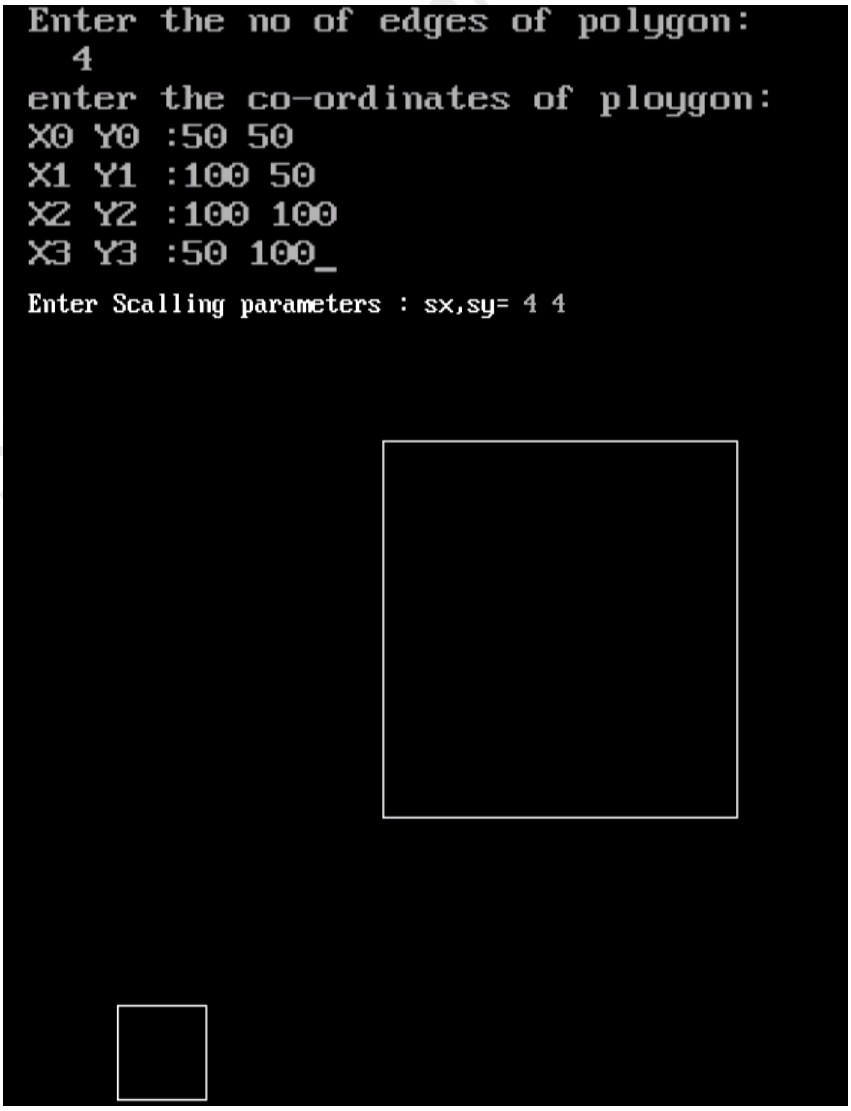


PRACTICAL-6(A)

Program for 2D-Scaling of Transformation

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<maths.h>
void main()
{
int x,y,x1,y1,x2,y2,xmax,ymax;
int gd=DETECT,gm,n,i,tx,ty,a[20][2],b[20][2];    float sx,sy;
clrscr();
cout<<"Enter the no of edges of polygon:\n  ";
cin>>n;
cout<<"enter the co-ordinates of ploygon: \n";
for(i=0;i<n;i++)
{
cout<<"X"<<i<<" Y"<<i<<" :";
cin>>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:\\tc\\bgi");
xmax=640;
ymax=480;
for(i=0;i<n;i++)
{
x1=a[i][0];
y1=ymax-a[i][1];
x2=a[i+1][0];
y2=ymax-a[i+1][1];
line(x1,y1,x2,y2);
}
cout<<"Enter Scalling parameters : sx,sy= ";
cin>>sx>>sy;
for(i=0;i<n;i++)
{
b[i][0]=(int)a[i][0]*sx;
b[i][1]=(int)a[i][1]*sy;
}
b[i][0]=b[0][0];
b[i][1]=b[0][1];
for(i=0;i<n;i++)
{
x1=b[i][0];
y1=ymax-b[i][1];
x2=b[i+1][0];
y2=ymax-b[i+1][1];
```

OUTPUT



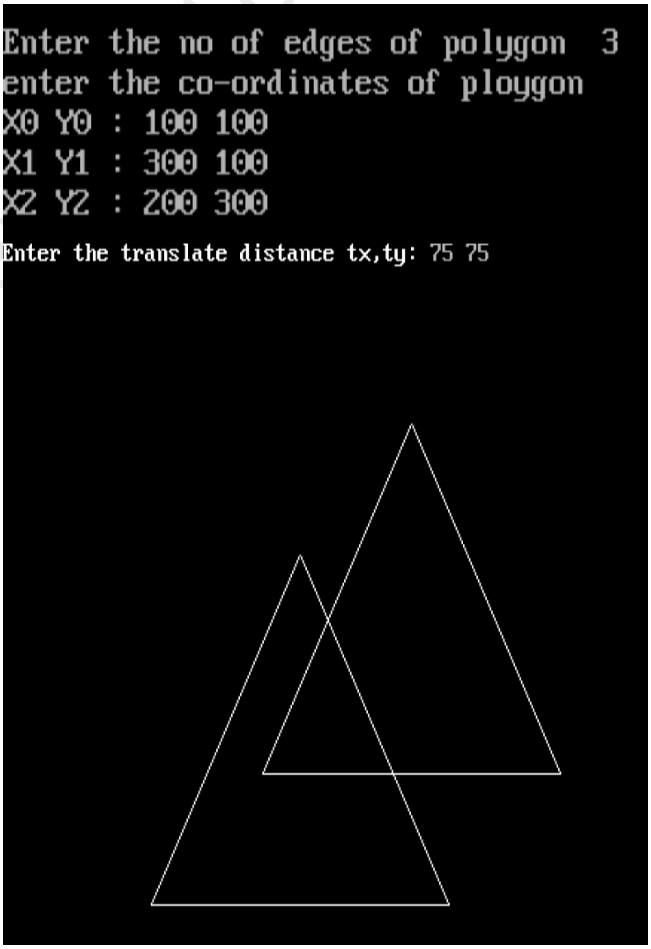
```
line(x1,y1,x2,y2);
}   getch();   }
```

PRACTICAL-6(B)

Program for 2D-Translation of Transformation

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
int x,y,x1,y1,x2,y2,xmax,ymax;
int gd=DETECT,gm,n,i,tx,ty,a[20][2],b[20][2];
clrscr();
cout<<"Enter the no of edges of polygon";
cin>>n;
cout<<"enter the co-ordinates of ploygon";
for(i=0;i<n;i++)
{
cout<<"X"<<i<<" Y"<<i<<" : ";
cin>>a[i][0]>>a[i][1];
}
a[n][0]=a[0][0];
a[n][1]=a[0][1];
initgraph(&gd,&gm,"c:\\tc\\bgi");
xmax=640;
ymax=480;
for(i=0;i<n;i++)
{
x1=a[i][0];
y1=ymax-a[i][1];
x2=a[i+1][0];
y2=ymax-a[i+1][1];
line(x1,y1,x2,y2);
}
cout<<"Enter the translate distance tx,ty: ";
cin>>tx>>ty;
for(i=0;i<n;i++)
{
b[i][0]=a[i][0]+tx;
b[i][1]=a[i][1]+ty;
}
b[n][0]=b[0][0];
b[n][1]=b[0][1];
for(i=0;i<n;i++)
{
x1=b[i][0];
y1=ymax-b[i][1];
x2=b[i+1][0];
y2=ymax-b[i+1][1];
line(x1,y1,x2,y2);
}
```

OUTPUT



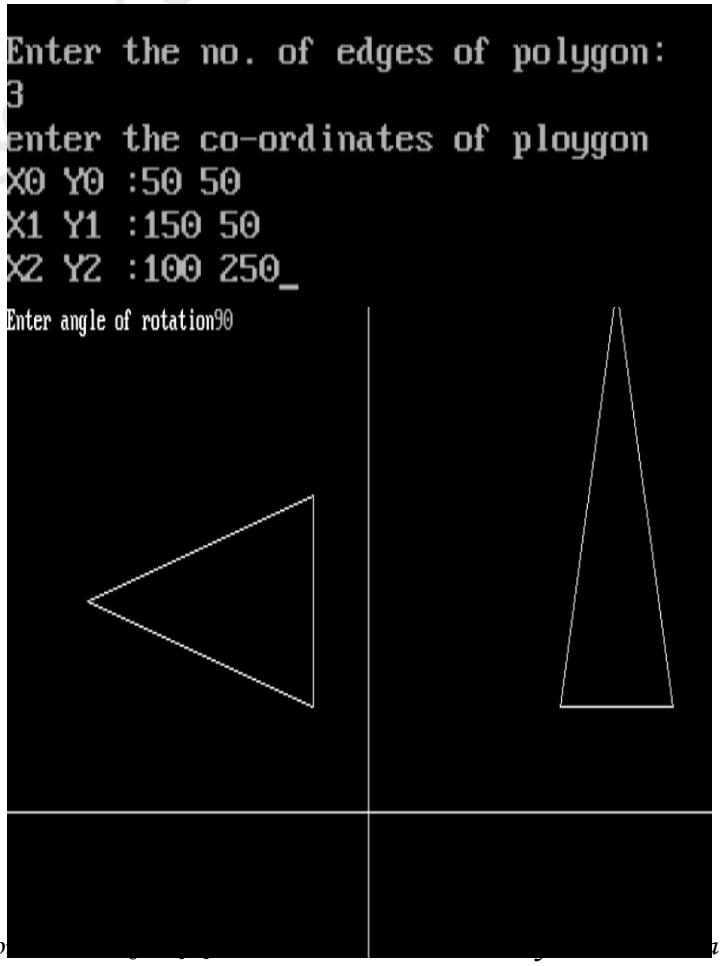
```
getch();
}
```

PRACTICAL-7

Write a program for 2-D ROTATION of TRANSFORMATION.

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
int x,y,x1,y1,x2,y2,xmax,ymax;
int gd=DETECT,gm,n,i,a[20][2],b[20][2];
float theta;
clrscr();
cout<<"Enter the no. of edges of polygon:\n";
cin>>n;
cout<<"enter the co-ordinates of ploygon \n";
for(i=0;i<n;i++)
{
cout<<"X"<<i<<" Y"<<i<<" :";
cin>>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:\\tc\\bgi");
line(320,0,320,480);
line(0,240,640,240);
x=320;
y=240;
xmax=640;
ymax=480;
for(i=0;i<n;i++)
{
x1=xmax-a[i][0];
y1=ymax-(y+a[i][1]);
x2=xmax-a[i+1][0];
y2=ymax-(y+a[i+1][1]);
```

OUTPUT



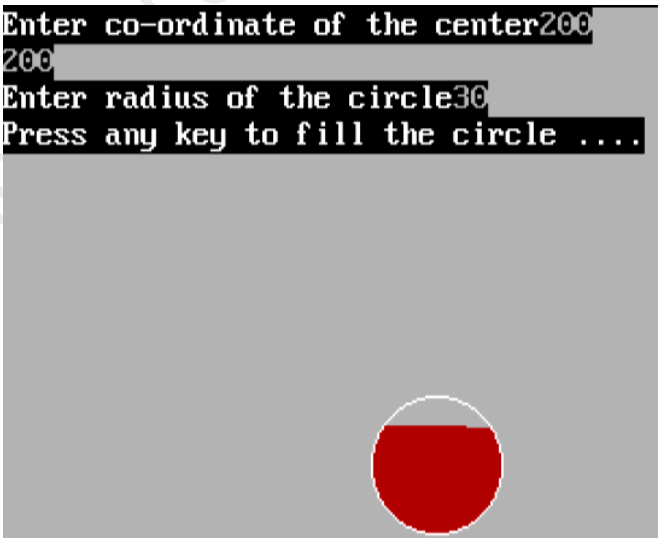
```
line(x1,y1,x2,y2);
}



cout<<"Enter angle of rotation";
cin>>theta;
theta=(theta*3.14)/180;
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[n][0]=b[0][0];
b[n][1]=b[0][1];
for(i=0;i<n;i++)
{
x1=x+b[i][0];
y1=ymax-(y+b[i][1]);
x2=x+b[i+1][0];
y2=ymax-(y+b[i+1][1]);
line(x1,y1,x2,y2);
}
getch();
}
```

PRACTICAL-9(B)

Write a program to fill a circle using Boundary Fill Algorithm.

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void bfill(int x,int y,int f_col,int b_col)
{
 int current=getpixel(x,y);
 if(current!=f_col&&current!=b_col)
 {
 delay(10);
 putpixel(x,y,f_col);
 bfill(x+1,y,f_col,b_col);
 bfill(x-1,y,f_col,b_col);
 bfill(x,y+1,f_col,b_col);
 bfill(x,y-1,f_col,b_col);
 }
 }
 void main()
 {
        int gd=DETECT,gm,xc,yc,r;
        initgraph(&gd,&gm,"c:\\tc\\bgi");
        clrscr();
        cout<<"Enter co-ordinate of the center";
        cin>>xc>>yc;
        cout<<"Enter radius of the circle";
        cin>>r;
        circle(xc,yc,r);
        cout<<"Press any key to fill the circle ....";
    //  getch();
        bfill(xc,yc,RED,WHITE);
        getch();
        closegraph();
```

OUTPUT



```
}
```

NOTE: Don't put radius value more than 40.

PRACTICAL-10(A)

Develop a simple text screen saver using graphics Algorithm.

```
#include<stdio.h>
#include<stdlib.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>
void main()
{
int gd=DETECT,gm,xmax=640,ymax=480,font=4,direction=2,size=8,color=15;
initgraph(&gd,&gm,"C:\\TC\\BGI");
cleardevice();
while(!kbhit())
  {
  settextstyle(random(font),random(direction),random(size));
   setcolor(random(color));
   outtextxy(random(ymax),random(xmax),"Vikas College");
   delay(200);
   }
}
```

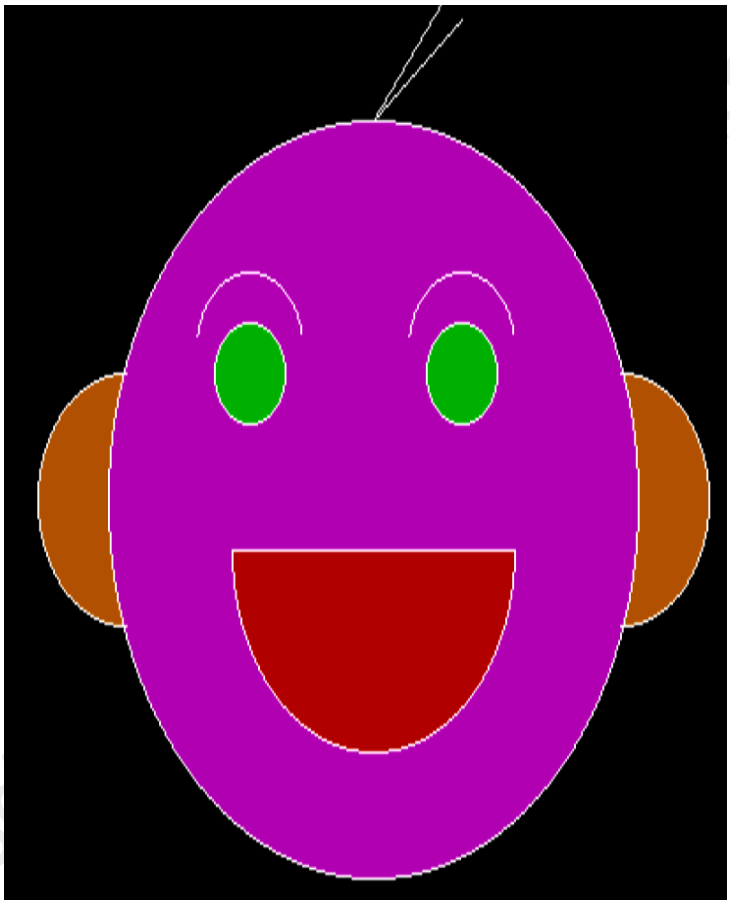
PRACTICAL-10(B)

Perform Smiling face animation using graphic function.

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
 {
int gd=DETECT,gm,i;
initgraph(&gd,&gm,"c:\\tc\\bgi");
//face
circle(250,250,150);
setfillstyle(1,5);
floodfill(225,225,15);
// Two Eyes
circle(180,200,20);
setfillstyle(1,2);
floodfill(181,200,15);
circle(300,200,20);
setfillstyle(1,2);
floodfill(300,200,15);
//mouth
arc(250,270,180,360,80);
line(170,270,330,270);
setfillstyle(1,4);
```

```
floodfill(255,275,15);


//eye brow
arc(180,190,10,170,30);
arc(300,190,10,170,30);
//Ear draw
arc(390,250,270,90,50);
setfillstyle(1,6);
floodfill(402,250,15);
arc(110,250,90,270,50);
setfillstyle(1,6);
floodfill(90,250,15);
//Hair
line(250,100,300,40);
line(250,100,300,60);
getch();
}
```

OUTPUT

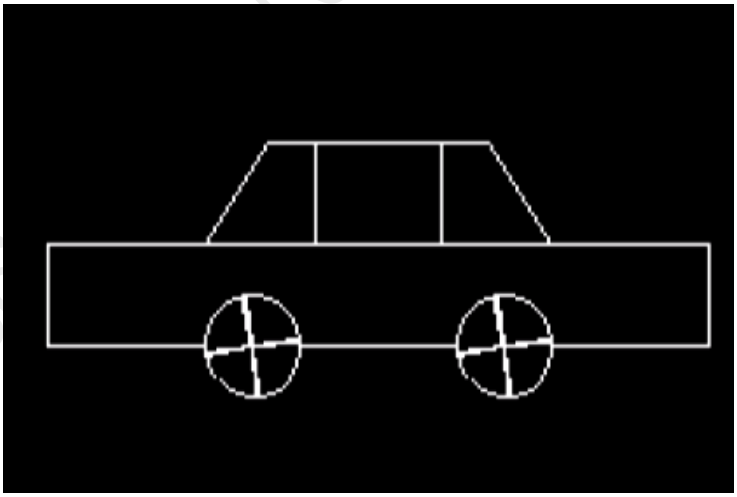


PRACTICAL-10(C)

Draw a moving car on the screen.

```
#include<graphics.h>
#include<conio.h>
#include<dos.h>
void main()
{
int gdriver=DETECT,gmode,i=0,j=0;
initgraph(&gdriver,&gmode,"c:\\tc\\bgi");
for(i;i<420;++i)
{
line(0+i,200,210+i,200);
line(50+i,200,70+i,170);
line(70+i,170,140+i,170);
line(140+i,170,160+i,200);
line(85+i,170,85+i,200);
line(125+i,170,125+i,200);
line(0+i,200,0+i,230);
line(210+i,200,210+i,230);
line(0+i,230,50+i,230);
circle(65+i,230,15);
line(80+i,230,130+i,230);
circle(145+i,230,15);
line(210+i,230,160+i,230);
pieslice(65+i,230,359-j,360-j,15);
pieslice(65+i,230,179-j,180-j,15);
pieslice(65+i,230,89-j,90-j,15);
pieslice(65+i,230,269-j,270-j,15);
pieslice(145+i,230,359-j,360-j,15);
pieslice(145+i,230,179-j,180-j,15);
pieslice(145+i,230,89-j,90-j,15);
pieslice(145+i,230,269-j,270-j,15);
if(j==179)
j=0;
++j;
delay(10);
cleardevice();
}
closegraph();
```

OUTPUT



```
}
```