ASSIGNMENT

OF

COMPUTER GRAPHICS

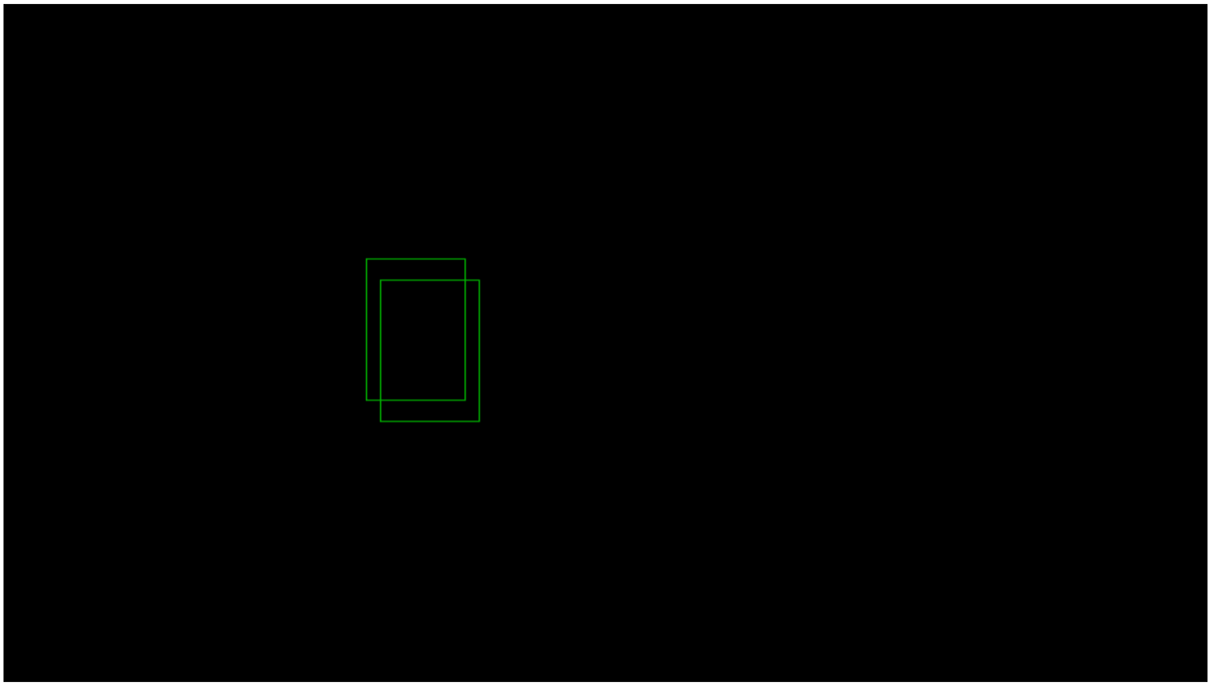
SUBMITTED TO:-	**SUBMITTED BY;-**
Prof.Balwinder kaur	**Saloni**
	Rollno.86
	MCA 4THsem
	(EVENING)

1)Program of translation.

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



// function to translate rectangle
void translateRectangle ( int P[][2], int T[])
{
        /* init graph and rectangle() are used for
        representing rectangle through graphical functions */
        int gd = DETECT, gm, errorcode;
        initgraph (&gd, &gm, "c:\\turboc3\\bgi");
        setcolor (2);
        // rectangle (Xmin, Ymin, Xmax, Ymax)
        // original rectangle
        rectangle (P[0][0], P[0][1], P[1][0], P[1][1]);

        // calculating translated coordinates
        P[0][0] = P[0][0] + T[0];
        P[0][1] = P[0][1] + T[1];
        P[1][0] = P[1][0] + T[0];
        P[1][1] = P[1][1] + T[1];

        // translated rectangle (Xmin, Ymin, Xmax, Ymax)
        // setcolor(3);
        rectangle (P[0][0], P[0][1], P[1][0], P[1][1]);
        // closegraph();
}

// driver program
```

```
int main()
{
        // Xmin, Ymin, Xmax, Ymax as rectangle
        // coordinates of top left and bottom right points
        int P[2][2] = {5, 8, 12, 18};
        int T[] = {2, 1}; // translation factor
        translateRectangle (P, T);
        return 0;
}
```

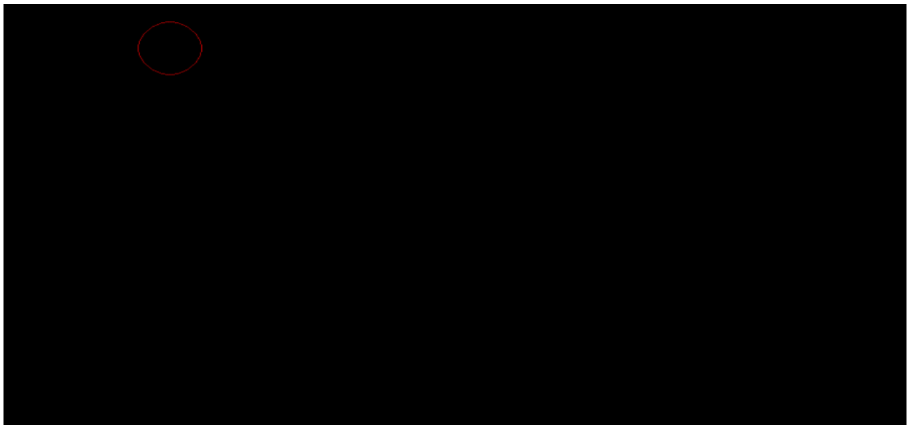
Output:-



2)Program to show concept of bresenham circle drawing algorithm.

```
#include <stdio.h>

#include <dos.h>

#include <graphics.h>


void drawCircle(int xc, int yc, int x, int y)

{

    putpixel(xc+x, yc+y, RED);

    putpixel(xc-x, yc+y, RED);

    putpixel(xc+x, yc-y, RED);

    putpixel(xc-x, yc-y, RED);

    putpixel(xc+y, yc+x, RED);

    putpixel(xc-y, yc+x, RED);

    putpixel(xc+y, yc-x, RED);

    putpixel(xc-y, yc-x, RED);

}


void circleBres(int xc, int yc, int r)

{

    int x = 0, y = r;

    int d = 3 - 2 * r;

    drawCircle(xc, yc, x, y);

    while (y >= x)

    {

// for each pixel we will

// draw all eight pixels


x++;


if (d > 0)
{
y--;
d = d + 4 * (x - y) + 10;
}
else
d = d + 4 * x + 6;
drawCircle(xc, yc, x, y);
delay(50);
   }
}



// driver function
void main()
{
   int xc = 50, yc = 50, r2 = 30;
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "c://turboc3//bgi");  // initialize graph
   circleBres(xc, yc, r2);    // function call
   getch();
}
```

Output:-



3)Program to show the concept of clipping.

```
#include<stdio.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;


printf("\nEnter x1 and y1\n");

scanf("%d %d",&p1.x,&p1.y);

printf("\nEnter x2 and y2\n");

scanf("%d %d",&p2.x,&p2.y);


initgraph(&gd,&gm,"c:\\turboc3\\bgi");
```

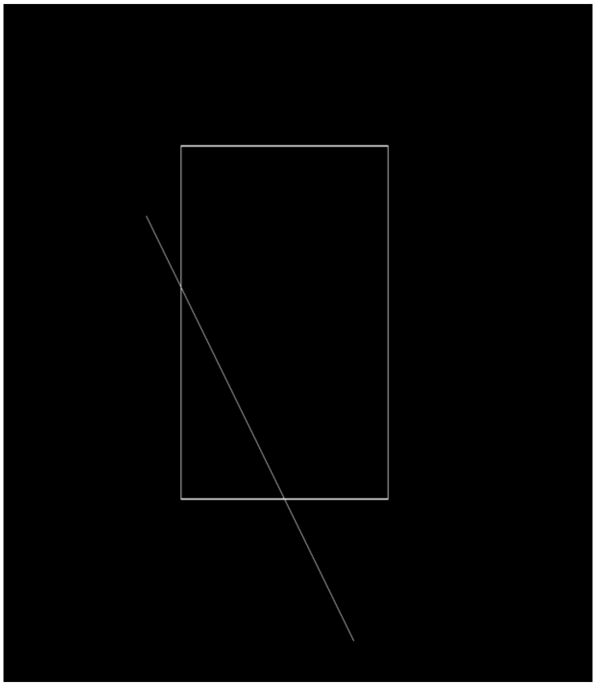
```
drawwindow();
delay(2000);

drawline(p1,p2);
delay(2000);
cleardevice();

delay(2000
);
p1=setcode(p1);
p2=setcode(p2);
v=visibility(p1,p2);
delay(2000);

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;
}
```

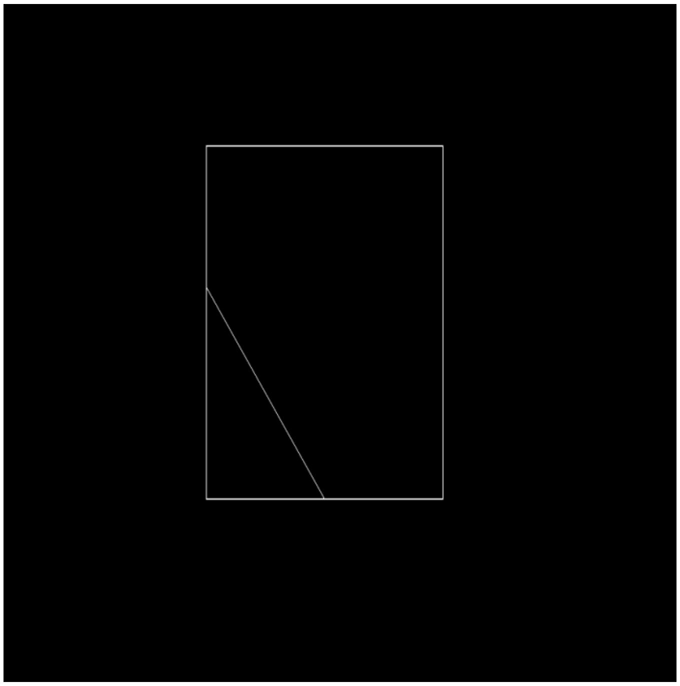
```
delay(5000);
closegraph();
}

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)
{
PT ptemp;

if(p.y<100)
ptemp.code[0]='1';
else
ptemp.code[0]='0';

if(p.y>350)
ptemp.code[1]='1';
else
ptemp.code[1]='0';
```

```
if(p.x>450)
ptemp.code[2]='1';
else
ptemp.code[2]='0';

if(p.x<150)
ptemp.code[3]='1';
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:-

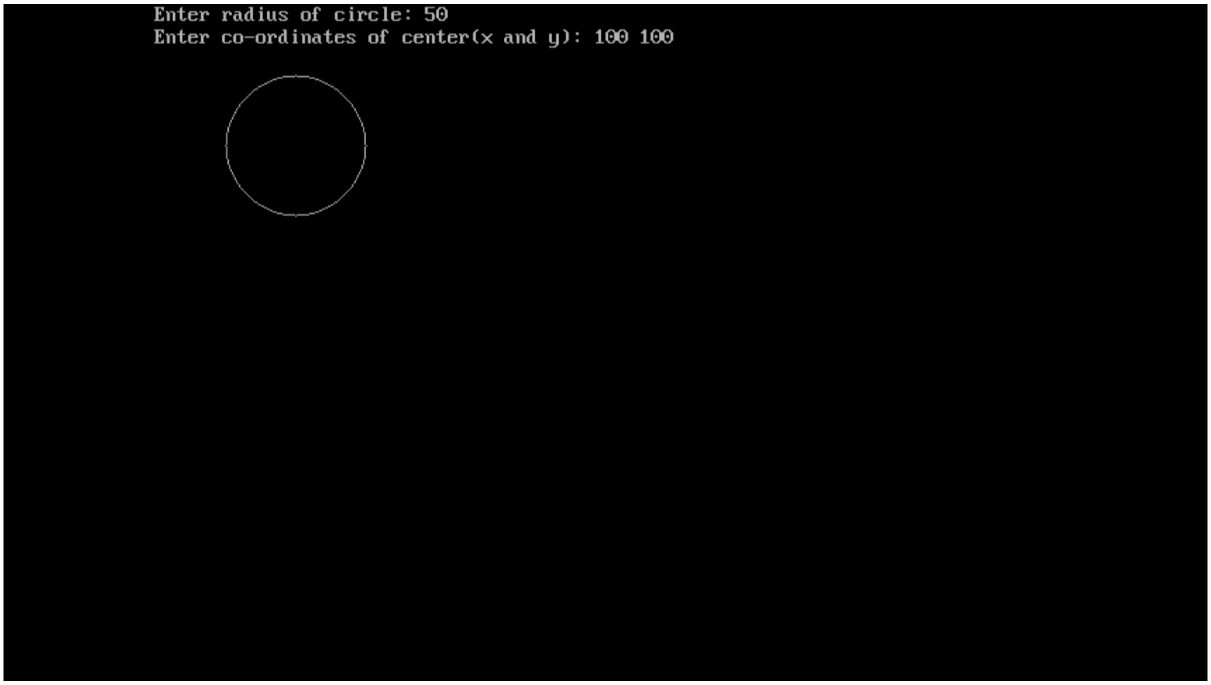




4)Program to show the concept of mid point algorithm

```
#include<stdio.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;
    }
    }
}
 void main()
{
    int gdriver=DETECT, gmode, error, x, y, r;
    initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
           printf("Enter radius of circle: ");
    scanf("%d", &r);

    printf("Enter co-ordinates of center(x and y): ");
    scanf("%d%d", &x, &y);
    drawcircle(x, y, r);
    getch();
}
```

Output:-



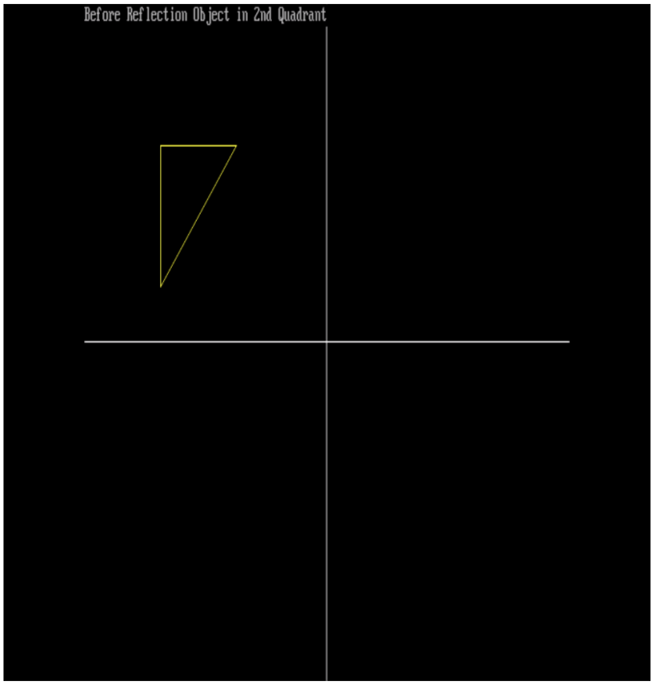
```
Enter radius of circle: 50
Enter co-ordinates of center(x and y): 100 100
```

5)Program to show reflection transformation.

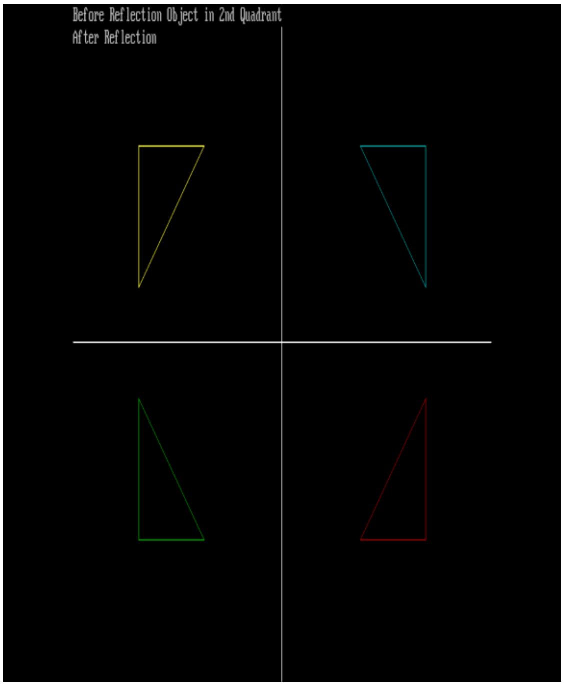
```
// C program for the above approach


#include <conio.h>

#include <graphics.h>

#include <stdio.h>


// Driver Code

void main()

{

// Initialize the drivers

int gm, gd = DETECT, ax, x1 = 100;

int x2 = 100, x3 = 200, y1 = 100;

int y2 = 200, y3 = 100;


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


cleardevice();
```

```
// Draw the graph
line(getmaxx() / 2, 0, getmaxx() / 2,
getmaxy());
line(0, getmaxy() / 2, getmaxx(),
getmaxy() / 2);

// Object initially at 2nd quadrant
printf("Before Reflection Object"
" in 2nd Quadrant");

// Set the color
setcolor(14);
line(x1, y1, x2, y2);
line(x2, y2, x3, y3);
line(x3, y3, x1, y1);
getch();

// After reflection
printf("\nAfter Reflection");

// Reflection along origin i.e.,
// in 4th quadrant
setcolor(4);
line(getmaxx() - x1, getmaxy() - y1,
getmaxx() - x2, getmaxy() - y2);

line(getmaxx() - x2, getmaxy() - y2,
getmaxx() - x3, getmaxy() - y3);

line(getmaxx() - x3, getmaxy() - y3,
getmaxx() - x1, getmaxy() - y1);

// Reflection along x-axis i.e.,
// in 1st quadrant
setcolor(3);
line(getmaxx() - x1, y1,
getmaxx() - x2, y2);
line(getmaxx() - x2, y2,
getmaxx() - x3, y3);
line(getmaxx() - x3, y3,
getmaxx() - x1, y1);

// Reflection along y-axis i.e.,
// in 3rd quadrant
setcolor(2);
line(x1, getmaxy() - y1, x2,
getmaxy() - y2);
line(x2, getmaxy() - y2, x3,
getmaxy() - y3);
line(x3, getmaxy() - y3, x1,
getmaxy() - y1);
getch();

// Close the graphics
closegraph();
}
```

Output:-



Before Reflection Object in 2nd Quadrant



Before Reflection Object in 2nd Quadrant
After Reflection

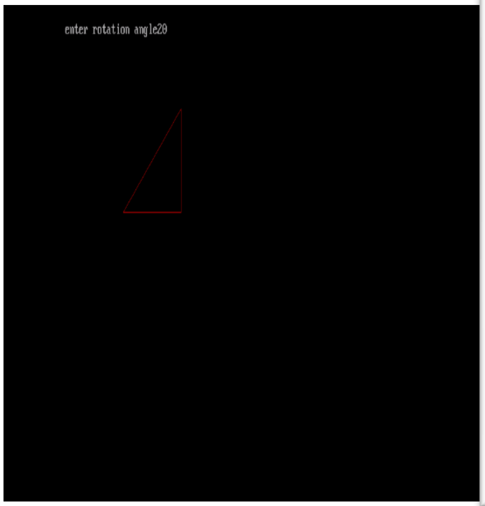
6)Program to show rotation transformation.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int x1,y1,x2,y2,x3,y3,a,b;
void draw();
void rotate();
int main(void)
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
printf("Enter first co-ordinate value for triangle:");
scanf("%d%d",&x1,&y1);
printf("Enter second co-ordinatevalues for triangle:");
scanf("%d%d",&x2,&y2);
printf("Enter third co-ordinate valuesfor triangle:");
scanf("%d%d",&x3,&y3);
draw();
getch();
```

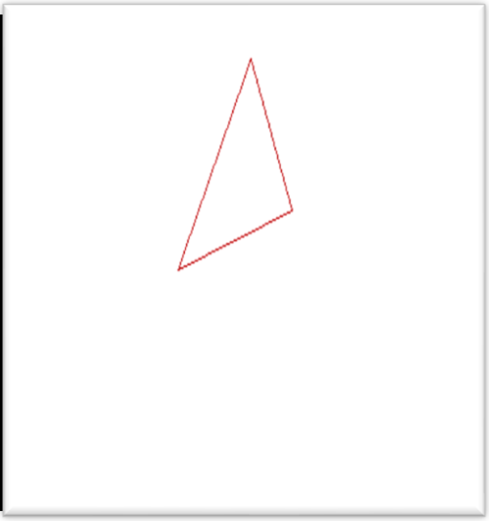
```
rotate();
getch();

return 0;
}

void draw()
{
  line(x1,y1,x2,y2);
  line(x2,y2,x3,y3);
  line(x3,y3,x1,y1);
}
 void rotate()
 {
    int a1,a2,a3,b1,b2,b3;
    float angle;
    printf("Enter the rotation angle co-ordinates:");
    scanf("%f",&angle);
    cleardevice();
      angle=(angle*3.14)/180;
      a1=a+(x1-a)*cos(angle)-(y1-b)*sin(angle);
      b1=b+(x1-a)*sin(angle)+(y2-b)*cos(angle);
      a2=a+(x2-a)*cos(angle)-(y1-b)*sin(angle);
      b2=b+(x2-a)*sin(angle)+(y2-b)*cos(angle);
      a3=a+(x3-a)*cos(angle)-(y1-b)*sin(angle);
      b3=b+(x3-a)*sin(angle)+(y2-b)*cos(angle);
      printf("ROTATION");
      printf("\n Changed coordinates\n");
      printf("%d %d\n%d %d\n%d %d",a1,b1,a2,b2,a3,b3);
    line(a1,b1,a2,b2);
    line(a2,b2,a3,b3);
```

```
   line(a3,b3,a1,b1);
 }
```

Output:-



```
enter rotation angle20
```



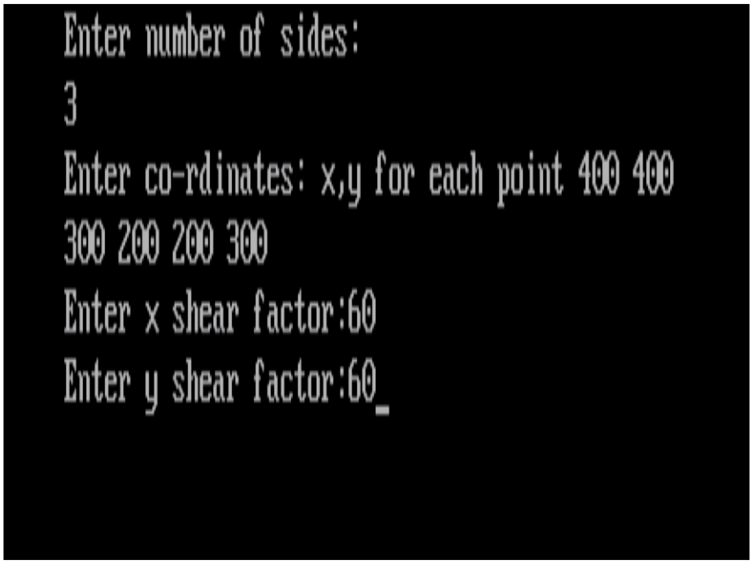
7)Program to show shearing transformation.

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
int gd=DETECT,gm;
int n,xs[100],ys[100],i;
float shearXfactor,shearYfactor;

void shear1()
{
for(i=0;i<n;i++)
 line(xs[i],ys[i],xs[(i+1)%n],ys[(i+1)%n]);
}

void shearAlongX()
{
for(i=0;i<n;i++)
```

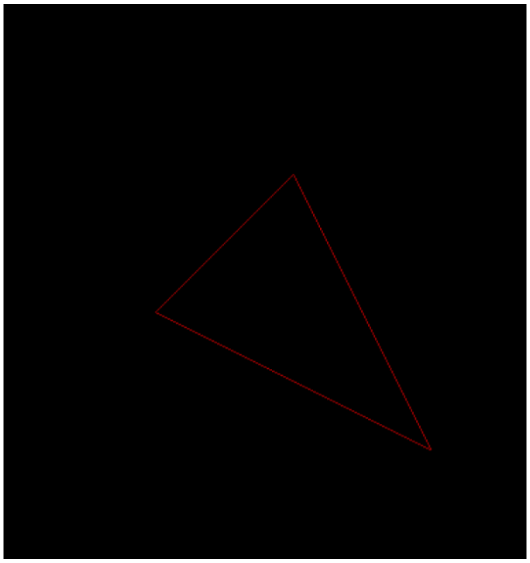
```
 xs[i]=xs[i]+shearXfactor*ys[i];
}

void shearAlongY()
{
for(i=0;i<n;i++)
 ys[i]=ys[i]+shearYfactor*xs[i];
}

void main()
{
printf("Enter number of sides: ");
scanf("%d",&n);
printf("Enter co-rdinates: x,y for each point ");
for(i=0;i<n;i++)
 scanf("%d%d",&xs[i],&ys[i]);
printf("Enter x shear factor:");
scanf("%f",&shearXfactor);
printf("Enter y shear factor:");
scanf("%f",&shearYfactor);

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI\\");
setcolor(RED);
shear1();//original
shearAlongX();
setcolor(BLUE);
shear1();//Xshear
shearAlongY();
setcolor(GREEN);
shear1();//Yshear
getch();
```

}

Output:-



```
Enter number of sides:
3
Enter co-rdinates: x,y for each point 400 400
300 200 200 300
Enter x shear factor:60
Enter y shear factor:60
```

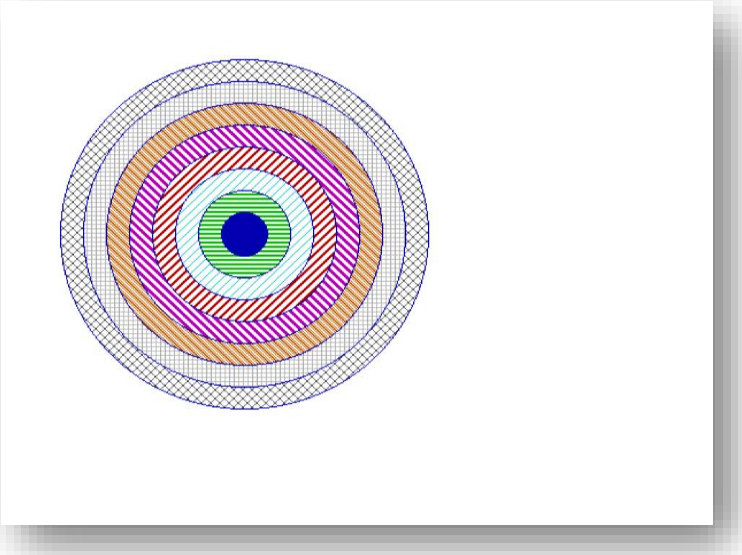


8)program of concentric circles.

```
#include<graphics.h>
#include<conio.h>
void main()
{
   int gd=DETECT, gm, i, x, y;
   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
   x=getmaxx()/3;
   y=getmaxx()/3;
   setbkcolor(WHITE);
   setcolor(BLUE);
for(i=1;i<=8;i++)
{
setfillstyle(i,i);
delay(20);
circle(x, y, i*20);
floodfill(x-2+i*20,y,BLUE);
```

```
    }
    getch();
    closegraph();
}
```

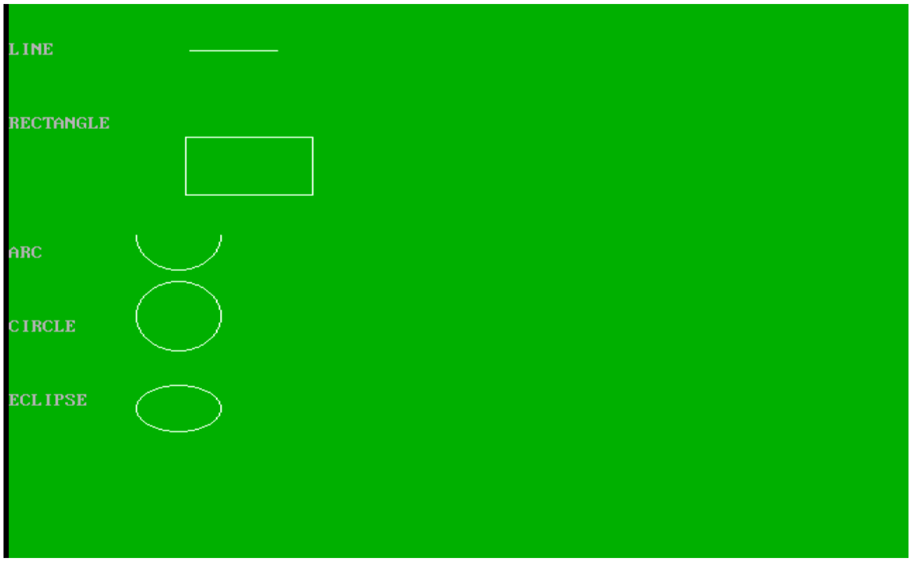
Output:-



9)Program to draw all shapes using inbuilt functions.

```
#include<graphics.h>
#include<conio.h>
void main()
{
    int gd=DETECT,gm;
    initgraph (&gd,&gm,"c:\\turboc3\\bgi");
    setbkcolor(GREEN);
    printf("\t\t\t\n\nLINE");
    line(50,40,190,40);
    printf("\t\t\n\n\n\nRECTANGLE");
    rectangle(125,115,215,165);
    printf("\t\t\t\n\n\n\n\n\n\nARC");
    arc(120,200,180,0,30);
    printf("\t\n\n\n\nCIRCLE");
    circle(120,270,30);
    printf("\t\n\n\n\nECLIPSE");
    ellipse(120,350,0,360,30,20);
    getch();
}
```

Output:-

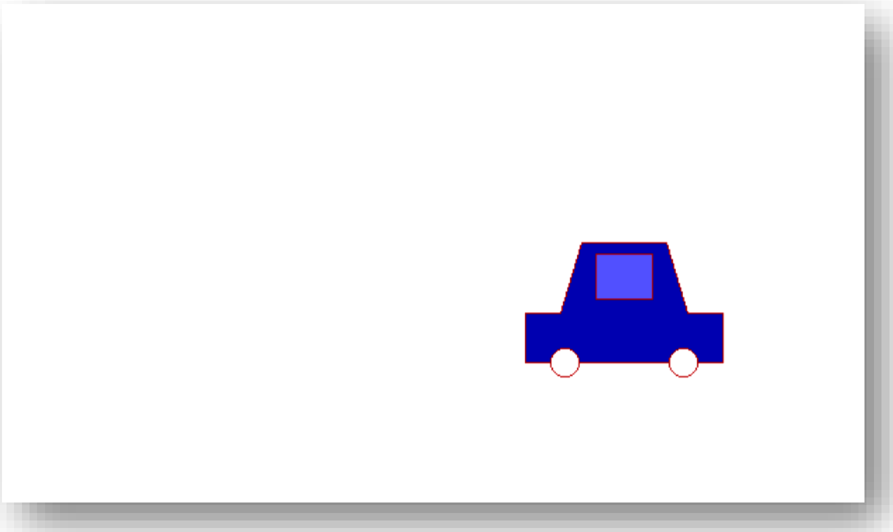


```
LINE

RECTANGLE


ARC

CIRCLE

ECLIPSE
```

10)Program of moving car.

```
#include<graphics.h>
#include<conio.h>
int main()
{
    int gd=DETECT,gm, i, maxx, cy;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
    setbkcolor(WHITE);
    setcolor(RED);
    maxx = getmaxx();
    cy = getmaxy()/2;
    for(i=0;i<maxx-140;i++)
        {
        cleardevice();
        line(0+i,cy-20, 0+i, cy+15);
        line(0+i, cy-20, 25+i, cy-20);
        line(25+i, cy-20, 40+i, cy-70);
        line(40+i, cy-70, 100+i, cy-70);
        line(100+i, cy-70, 115+i, cy-20);
        line(115+i, cy-20, 140+i, cy-20);
        line(0+i, cy+15, 18+i, cy+15);
        circle(28+i, cy+15, 10);
        line(38+i, cy+15, 102+i, cy+15);
        circle(112+i, cy+15,10);
        line(122+i, cy+15 ,140+i,cy+15);
        line(140+i, cy+15, 140+i, cy-20);
        rectangle(50+i, cy-62, 90+i, cy-30);
        setfillstyle(1,BLUE);
        floodfill(5+i, cy-15, RED);
        setfillstyle(1, LIGHTBLUE);
        floodfill(52+i, cy-60, RED);
        delay(10);
         }
    getch();
    closegraph();
    return 0;
}
```

Output:-

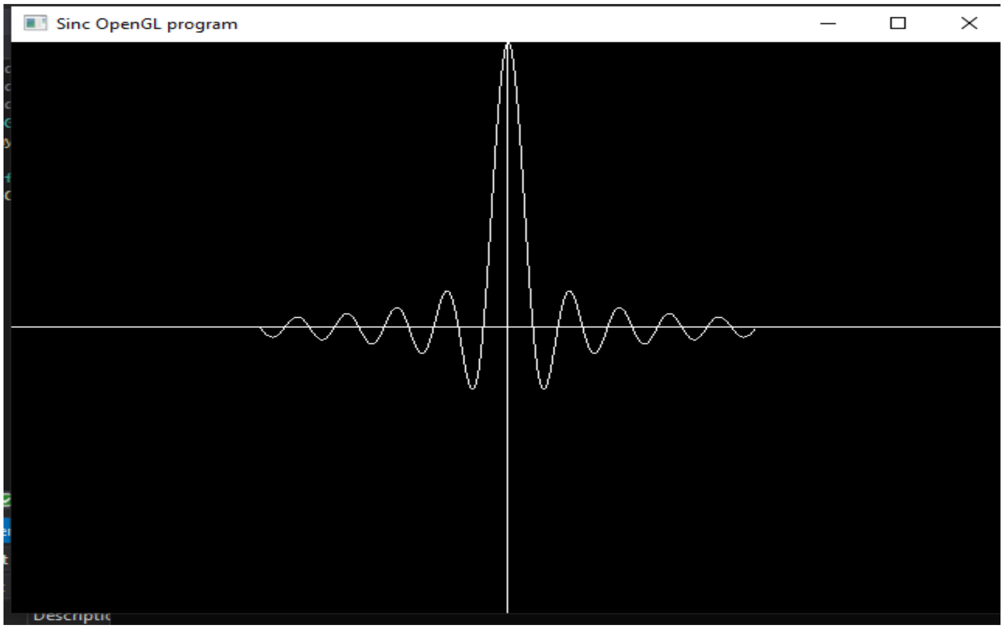


11)Program of Sine Curve in opengl.

```
#include<iostream>
#include<Gl/glut.h>
#include<math.h>
const GLfloat factor = 0.2f;
void myDisplay(void)
{
    GLfloat x;
    glClear(GL_COLOR_BUFFER_BIT);
    {
        glColor3f(1.0f, 1.0f, 1.0f);
        glBegin(GL_LINES);
        {
            // x-axis
            glVertex3f(-100.0f, 0.0f, 0.0f);
            glVertex3f(100.0f, 0.0f, 0.0f);
            // y-axis
            glVertex3f(0.0f, -100.0f, 0.0f);
            glVertex3f(0.0f, 100.0f, 0.0f);
            glEnd();
            glBegin(GL_LINE_STRIP);
            for (x = -2.0f / factor; x < 2.0f / factor; x += 0.030f)
            {
                glVertex2f((x * factor) / 4, sin(3.14159 * x) / (3.14159 * x));
            }
            glEnd();
            glFlush();
        }
    }
    glFlush();
    glutSwapBuffers();
}
```

```
int main(int argc, char* argv[])
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
        glutInitWindowPosition(100, 100);
        glutInitWindowSize(640, 480);
        glutCreateWindow("Sinc OpenGL program");

                glutDisplayFunc(&myDisplay);
        glutMainLoop();
        return 0;
}
```

Output:-

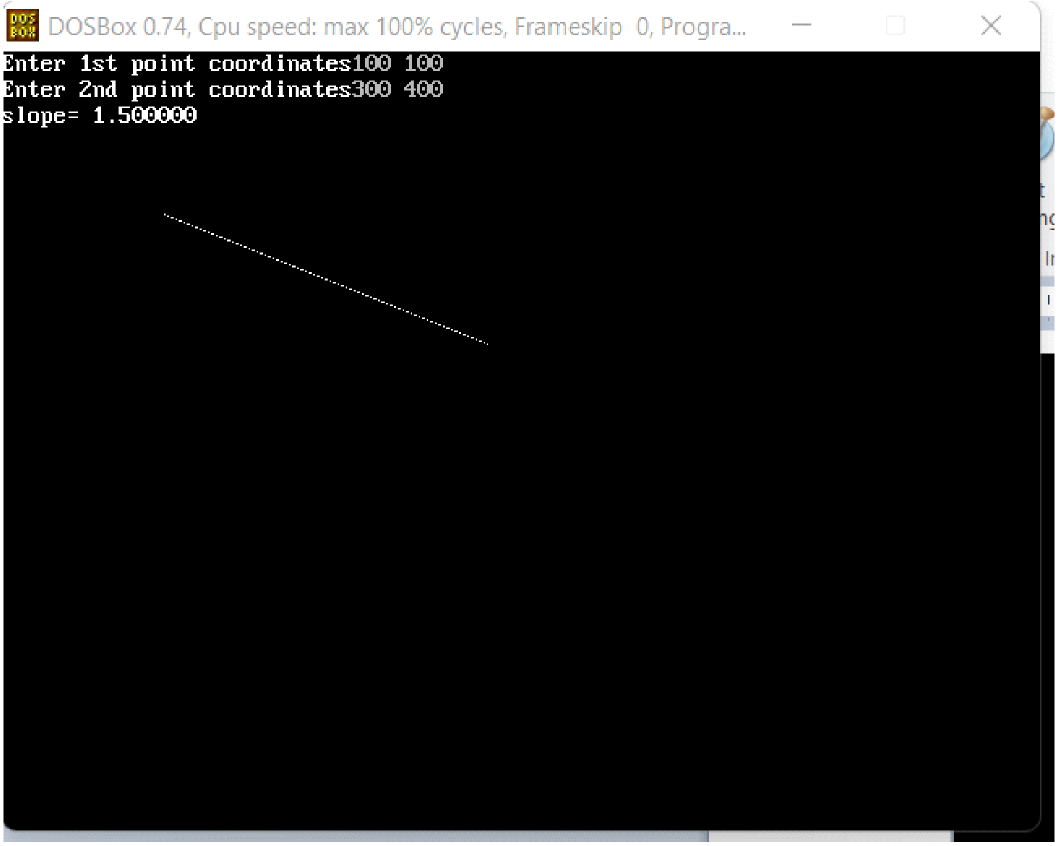


12)Program of bresenham line drawing algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
int main()
{
   int gd=DETECT, gm,x0, y0, x1, y1,dx,dy,p,x,y;
   float m;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
printf("Enter 1st point coordinates");
scanf("%d%d",&x0,&y0);
printf("Enter 2nd point coordinates");
scanf("%d%d",&x1,&y1);
   dx=x1-x0;
   dy=y1-y0;
```

```
    m=(float)dy/dx;
    printf("slope= %f",m,dx,dy);
    x=x0;
    y=y0;
    p=2*dy-dx;
if(m<1)
{
    while(x<x1)
    {
        if(p>0)
        {
            putpixel(x,y,WHITE);
            y=y+1;
            x= x+1;
            p=p+2*dy-2*dx;
        }
        else
        {
            putpixel(x,y,WHITE);
            p=p+2*dy;
        }
            x=x+1;
}
}
else
{
   while(x<x1)
   {
        if(p>0)
        {
                putpixel(x,y,WHITE);
                y=y+1;
                x=x+1;
                p=p+2*dx-2*dy;
        }
        else
        {
                putpixel(x,y,WHITE);
                p=p+2*dx;
        }
                x=x+1;
}
}
        getch();
        return 0;
        closegraph();
}
```

Output:-

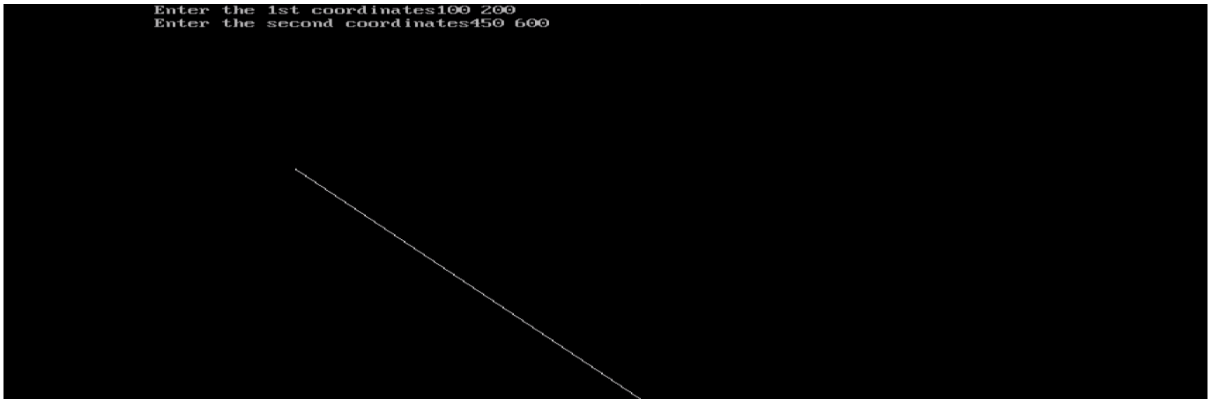

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

```
Enter 1st point coordinates100 100
Enter 2nd point coordinates300 400
slope= 1.500000
```

13)Program to show the concept of dda algorithm

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
int gd = DETECT,gm,i;
float x,y,dx,dy,steps;
int x0,x1,y0,y1;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("Enter the 1st coordinates");
scanf("%d%d",&x0,&y0);
printf("Enter the second coordinates");
scanf("%d%d",&x1,&y1);
dx = (float)(x1-x0);
dy = (float)(y1-y0);
if(dx>=dy)
      {
      steps = dx;
      }
else
      {
      steps = dy;
      }
dx = dx/steps;
dy = dy/steps;
x=x0;
y=y0;
i=1;
while(i<=steps)
{
      putpixel(x,y,WHITE);
      x += dx;
      y += dy;
      i = i+1;
}
getch();
closegraph();
}
```

Output :-



```
Enter the 1st coordinates100 200
Enter the second coordinates450 600
```

14)Progam to draw a flag.

#include<stdio.h>

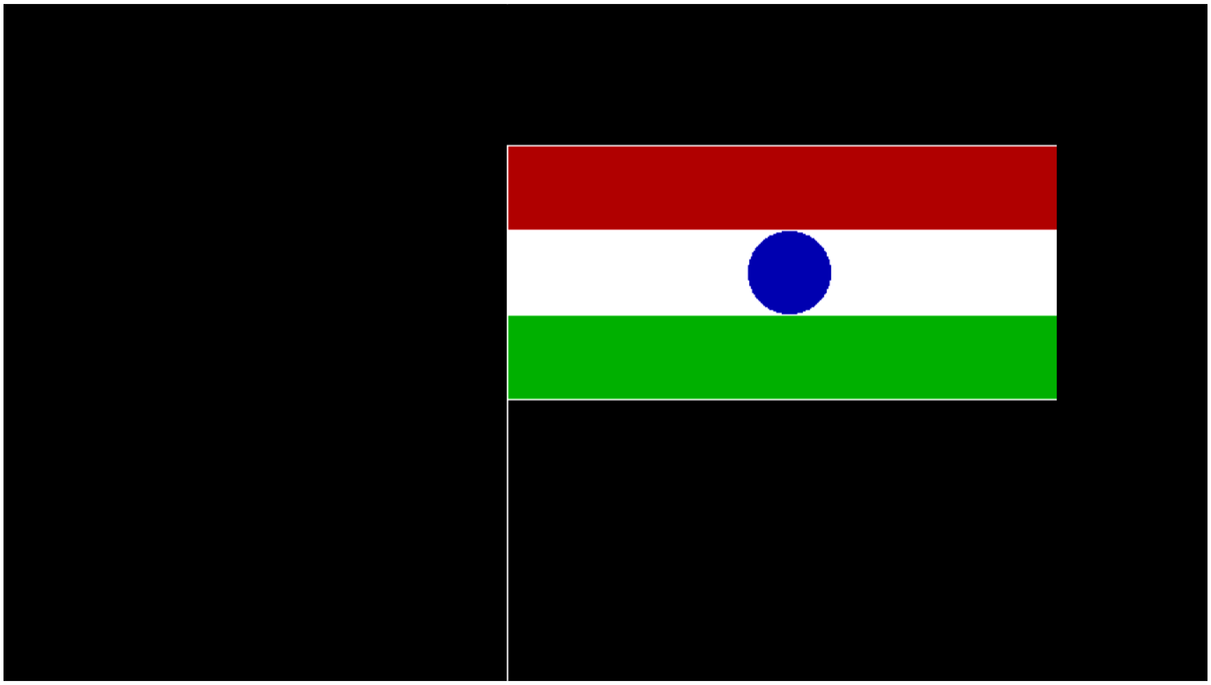
#include<conio.h>

#include<graphics.h>

void main()

```
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\Turboc3\\bgi");
line(250,100,250,600);
line(250,100,250,600);
setfillstyle(SOLID_FILL,WHITE);
rectangle(225,600,275,610);
rectangle(200,610,300,620);
floodfill(227,608,15);
floodfill(202,618,15);
setfillstyle(SOLID_FILL,RED);
rectangle(250,100,650,280);
line(250,160,650,160);
floodfill(252,158,15);
setfillstyle(SOLID_FILL,BLUE);
circle(450,190,30);
floodfill(452,188,15);
setfillstyle(SOLID_FILL,WHITE);
line(250,160,480,160);
line(250,220,480,220);
floodfill(252,162,15);
setfillstyle(SOLID_FILL,WHITE);
line(480,160,650,160);
line(480,220,650,220);
floodfill(482,162,15);
setfillstyle(SOLIpD_FILL,GREEN);
line(250,220,650,220);
floodfill(252,278,15);

getch();
closegraph();
}
```

Output :-

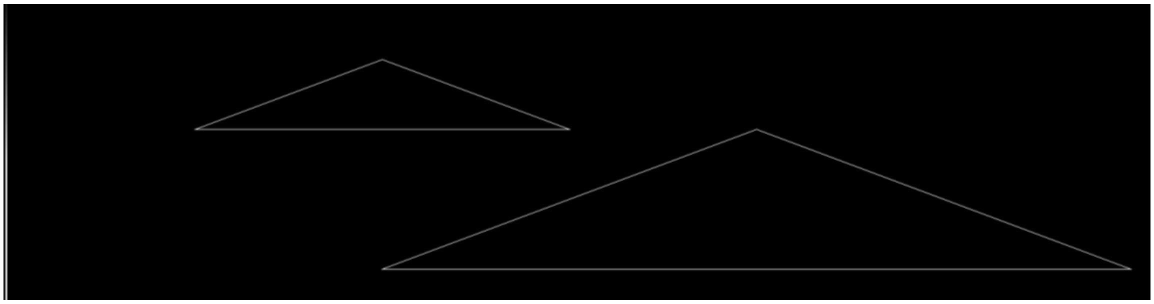


15)Program of Scaling.

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



void findNewCoordinate(int s[][2], int p[][1])
{
int temp[2][1] = { 0 };

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

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

// Scaling the Polygon
void scale(int x[], int y[], int sx, int sy)
{
// Triangle before Scaling
line(x[0], y[0], x[1], y[1]);
line(x[1], y[1], x[2], y[2]);
line(x[2], y[2], x[0], y[0]);

// Initializing the Scaling Matrix.
int s[2][2] = { sx, 0, 0, sy };
int p[2][1];

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

findNewCoordinate(s, p);

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

// Triangle after Scaling
line(x[0], y[0], x[1], y[1]);
line(x[1], y[1], x[2], y[2]);
line(x[2], y[2], x[0], y[0]);
}

// Driven Program
int main()
{
int x[] = { 100, 200, 300 };
int y[] = { 200, 100, 200 };
int sx = 2, sy = 2;

int gd=DETECT, gm;

initgraph(&gd, &gm,"c:\\TURBOC3\\bgi ");

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



}
```

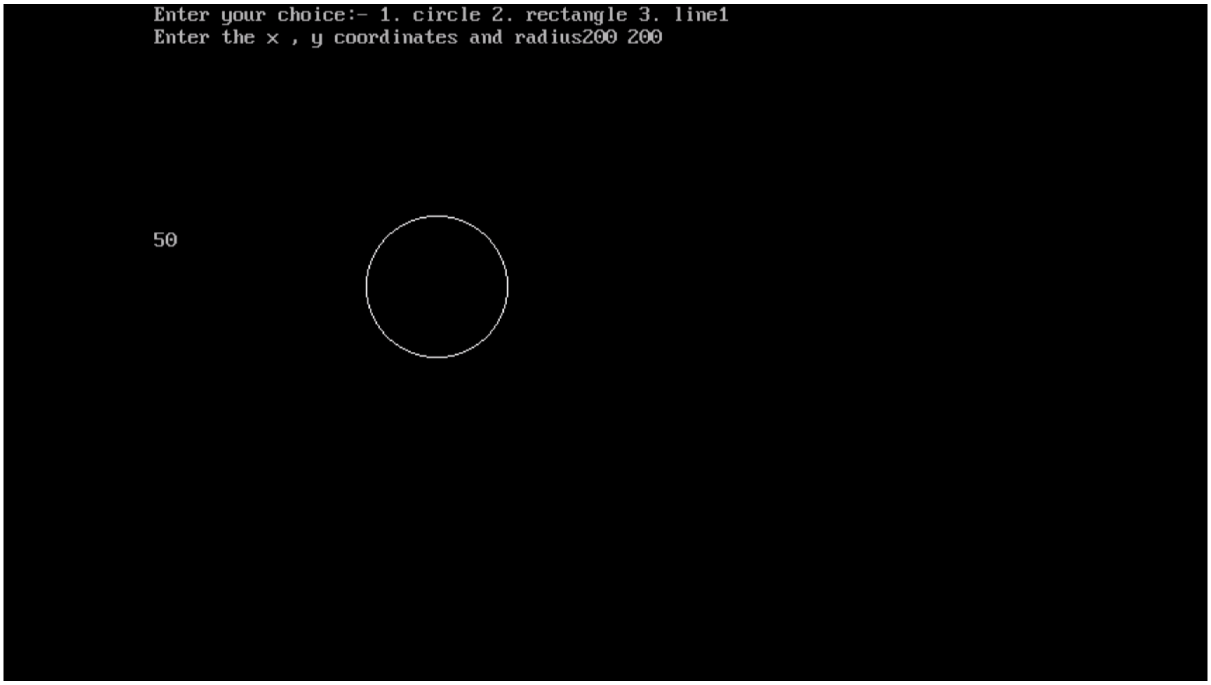
Output :-



16)Program of switch case.

```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>

int main()

{

int gd = DETECT,gm,n,x,y,r,x1,x2,y1,y2;

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

printf("Enter your choice:- 1. circle 2. rectangle 3. line");

scanf("%d",&n);

switch(n)

{

case 1 : printf("Enter the x , y coordinates and radius");

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

circle(x,y,r);

break;

case 2 : printf("Enter the coordinates for rectangle left upper right bottom");

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

rectangle(x1,y1,x2,y2);

break;

case 3 : printf("Enter the x and y coordinattes of a line");

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

line(x1,y1,x2,y2);

break;

default : printf("no option available");

}

getch();

closegraph();

return 0;

}
```

Output :-



```
Enter your choice:- 1. circle 2. rectangle 3. line1
Enter the x , y coordinates and radius200 200




50
```

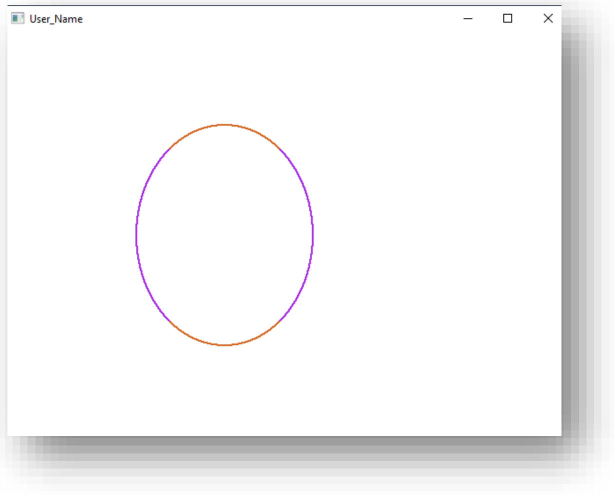
17)program of open gl to make an oval.

```
#include <GL/glut.h>

#include<iostream>

using namespace std;

int rx = 100, ry = 125;

int xCenter = 250, yCenter = 250;

void myinit(void)

{

        glClearColor(1.0, 1.0, 1.0, 0.0);

        glMatrixMode(GL_PROJECTION);

        glLoadIdentity();

        gluOrtho2D(0.0, 640.0, 0.0, 480.0);

}

void setPixel(GLint x, GLint y)

{
        glBegin(GL_POINTS);
        glVertex2i(x, y);
        glEnd();
}
void ellipseMidPoint()
{
        float x = 0;
        float y = ry;
        float p1 = ry * ry - (rx * rx) * ry + (rx * rx) * (0.25);
        float dx = 2 * (ry * ry) * x;
        float dy = 2 * (rx * rx) * y;
        glColor3ub(rand() % 255, rand() % 255, rand() % 255);
        while (dx < dy)
        {
            setPixel(xCenter + x, yCenter + y);
            setPixel(xCenter - x, yCenter + y);
            setPixel(xCenter + x, yCenter - y);
            setPixel(xCenter - x, yCenter - y);
            if (p1 < 0)
            {
                x = x + 1;
                dx = 2 * (ry * ry) * x;
                p1 = p1 + dx + (ry * ry);
            }
            else
            {
                x = x + 1;
```

```
                y = y - 1;
                dx = 2 * (ry * ry) * x;
                dy = 2 * (rx * rx) * y;
                p1 = p1 + dx - dy + (ry * ry);
        }
}
glFlush();
float p2 = (ry * ry) * (x + 0.5) * (x + 0.5) + (rx * rx) * (y
        - 1) * (y - 1) - (rx * rx) * (ry * ry);
glColor3ub(rand() % 255, rand() % 255, rand() % 255);
while (y > 0)
{
        setPixel(xCenter + x, yCenter + y);
        setPixel(xCenter - x, yCenter + y);
        setPixel(xCenter + x, yCenter - y);
        setPixel(xCenter - x, yCenter - y);
        if (p2 > 0)
        {
                x = x;
                y = y - 1;
                dy = 2 * (rx * rx) * y;
                p2 = p2 - dy + (rx * rx);
        }
        else
        {
                x = x + 1;
                y = y - 1;
```

```
            dy = dy - 2 * (rx * rx);
            dx = dx + 2 * (ry * ry);
            p2 = p2 + dx - dy + (rx * rx);
        }
    }
    glFlush();
}
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    glPointSize(2.0);
    ellipseMidPoint();
    glFlush();
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(10, 10);
    glutCreateWindow("User_Name");
    myinit();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

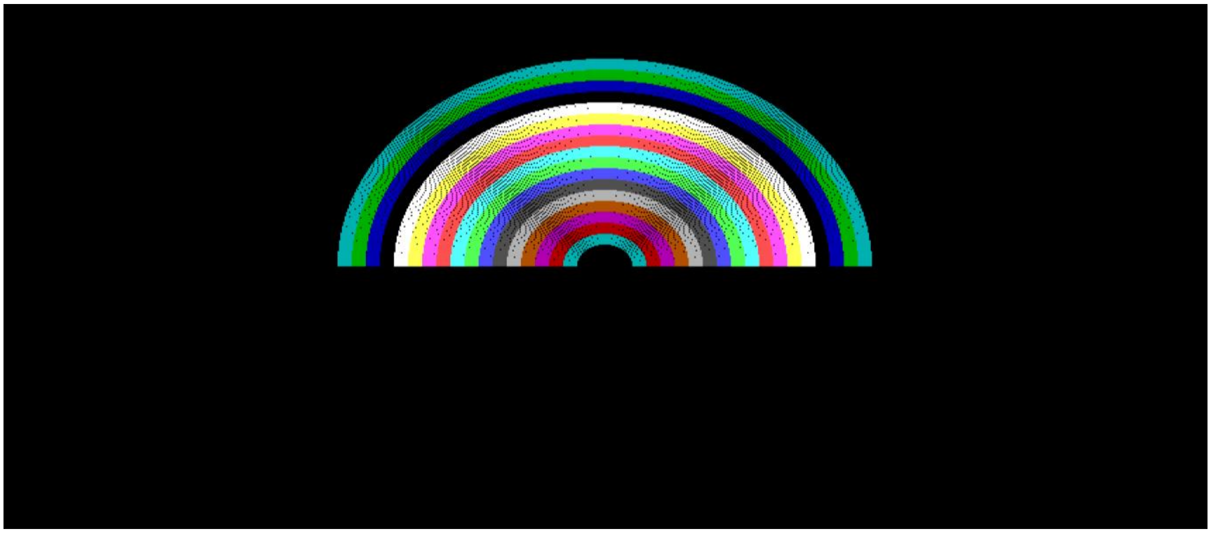
Output :-



18)program to make rainbow.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
int gdriver = DETECT,gmode;
int x,y,i;
        initgraph(&gdriver,&gmode,"C:\\Turboc3\\BGI");
        x=getmaxx()/2;
        y=getmaxy()/2;
        for(i=30;i<200;i++)
        {
                delay(100);
                setcolor(i/10);
                arc(x,y,0,180,i-10);
        }
getch();
}
```

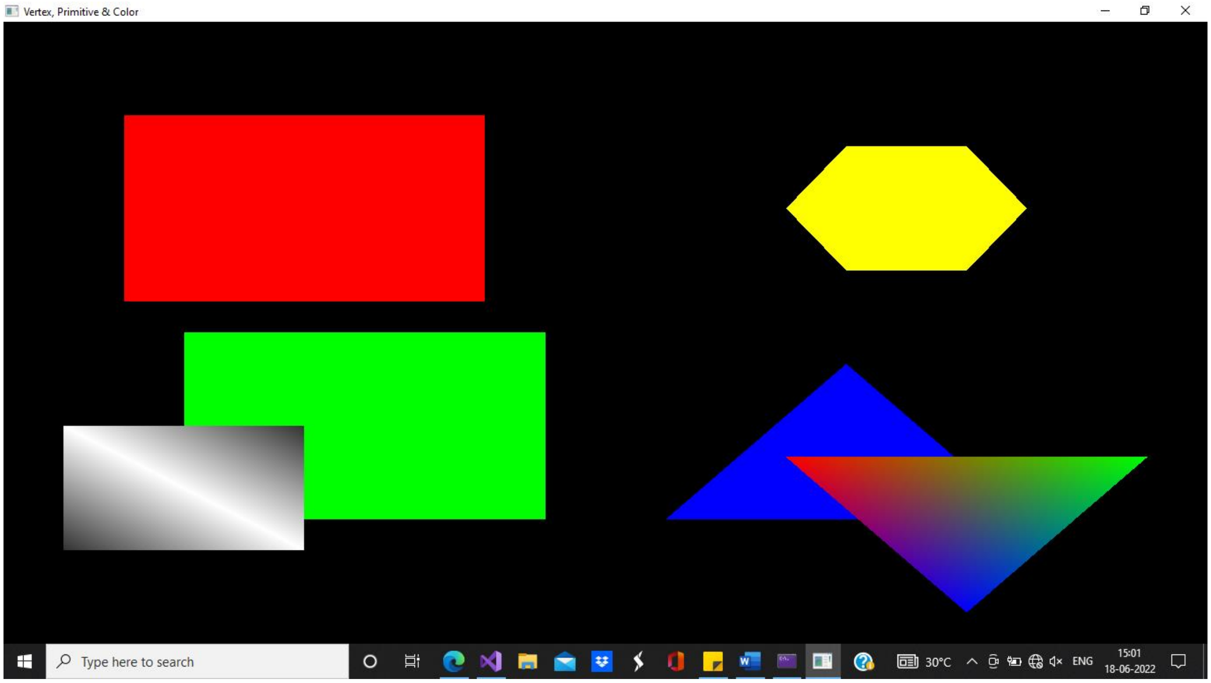
Output :-



19)open gl primitives

```
#include <windows.h>
#include <GL/glut.h>
void initGL() {

    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT);


    glBegin(GL_QUADS);
    glColor3f(1.0f, 0.0f, 0.0f); // Red
    glVertex2f(-0.8f, 0.1f);
    glVertex2f(-0.2f, 0.1f);
    glVertex2f(-0.2f, 0.7f);
    glVertex2f(-0.8f, 0.7f);

    glColor3f(0.0f, 1.0f, 0.0f); // Green
    glVertex2f(-0.7f, -0.6f);
    glVertex2f(-0.1f, -0.6f);
    glVertex2f(-0.1f, 0.0f);
    glVertex2f(-0.7f, 0.0f);

    glColor3f(0.2f, 0.2f, 0.2f); // Dark Gray
    glVertex2f(-0.9f, -0.7f);
    glColor3f(1.0f, 1.0f, 1.0f); // White
    glVertex2f(-0.5f, -0.7f);
    glColor3f(0.2f, 0.2f, 0.2f); // Dark Gray
    glVertex2f(-0.5f, -0.3f);
    glColor3f(1.0f, 1.0f, 1.0f); // White
    glVertex2f(-0.9f, -0.3f);
    glEnd();

    glBegin(GL_TRIANGLES);
    glColor3f(0.0f, 0.0f, 1.0f); // Blue
    glVertex2f(0.1f, -0.6f);
    glVertex2f(0.7f, -0.6f);
    glVertex2f(0.4f, -0.1f);

    glColor3f(1.0f, 0.0f, 0.0f); // Red
    glVertex2f(0.3f, -0.4f);
    glColor3f(0.0f, 1.0f, 0.0f); // Green
    glVertex2f(0.9f, -0.4f);
    glColor3f(0.0f, 0.0f, 1.0f); // Blue
    glVertex2f(0.6f, -0.9f);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3f(1.0f, 1.0f, 0.0f); // Yellow
    glVertex2f(0.4f, 0.2f);
    glVertex2f(0.6f, 0.2f);
    glVertex2f(0.7f, 0.4f);
    glVertex2f(0.6f, 0.6f);
    glVertex2f(0.4f, 0.6f);
    glVertex2f(0.3f, 0.4f);
    glEnd();

    glFlush();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutCreateWindow("Vertex, Primitive & Color);
    glutInitWindowSize(320, 320);
    glutInitWindowPosition(50, 50);
    glutDisplayFunc(display);
    initGL();
    glutMainLoop();
    return 0;
}
```

Output :-


Vertex, Primitive & Color
Type here to search
30°C
ENG
15:01
18-06-2022