

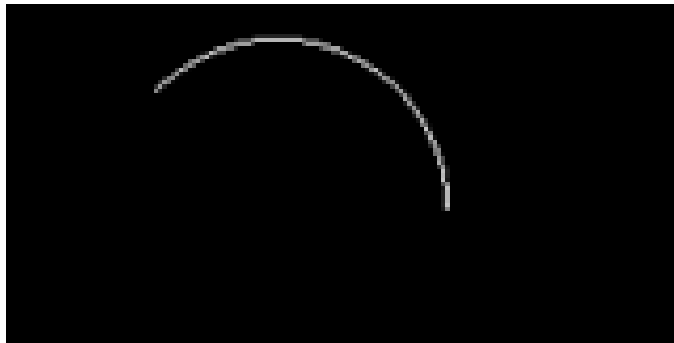
Practical – 1

Aim: Study and enlist the basic functions used for graphics in C language

Arc Function in C:

```
#include<graphics.h>
#include<conio.h>
void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    arc(100,100,0,135,50);
    getch();
    closegraph();
}
```

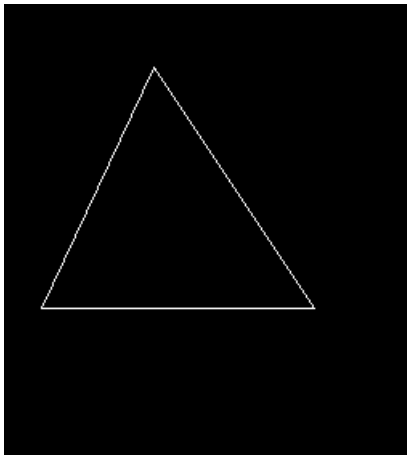
Output:



Drawpoly Function in C:

```
#include<graphics.h>
#include<conio.h>
void main()
{
    int gd=DETECT,gm,points[]={320,150,420,300,250,300,320,150};
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    drawpoly(4,points);
    getch();
    closegraph();
}
```

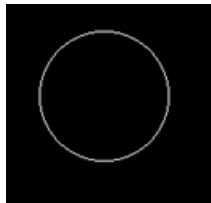
Output:



Circle Function in C:

```
#include<graphics.h>
#include<conio.h>
void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    circle(100,100,50);
    getch();
    closegraph();
}
```

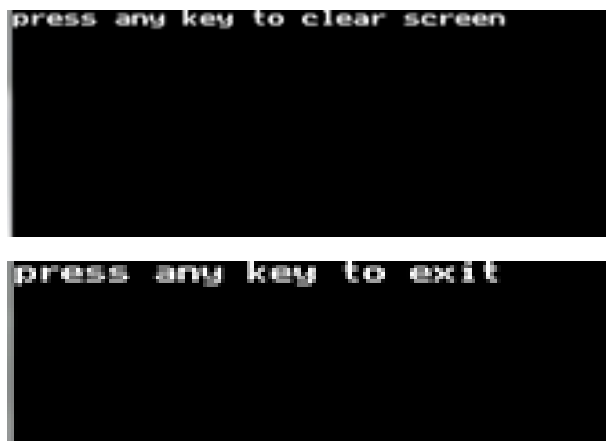
Output:



Cleardevice Function in C:

```
#include<graphics.h>
#include<conio.h>
void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    outtext("press any key to clear screen");
    getch();
    cleardevice();
    outtext("press any key to exit");
    getch();
    closegraph();
}
```

Output:



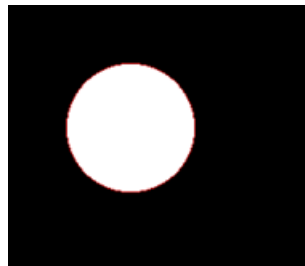
Floodfill Function in C:

```
#include<graphics.h>

#include<conio.h>

void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    setcolor(RED);
    circle(100,100,50);
    floodfill(100,100,RED);
    getch();
    closegraph();
}
```

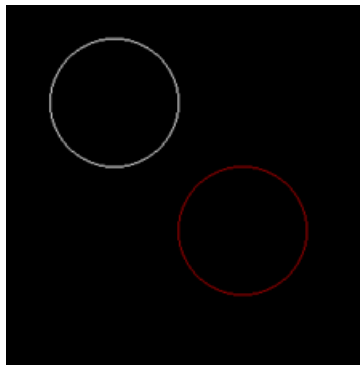
Output:



Setcolor Function in C:

```
#include<graphics.h>
#include<conio.h>
void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    circle(100,100,50);
    setcolor(RED);
    circle(200,200,50);
    getch();
    closegraph();
}
```

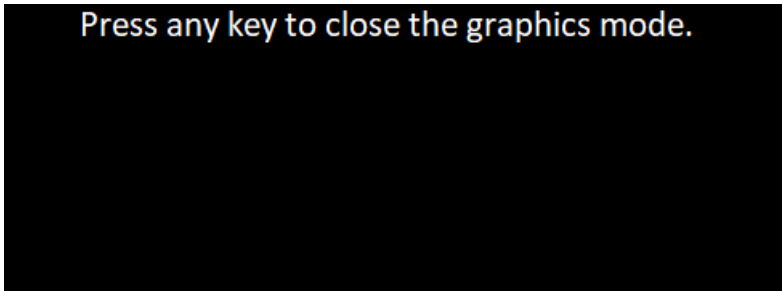
Output:



Drawpoly Function in C:

```
#include<graphics.h>
#include<conio.h>
void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    outtext("Press any key to close the graphics mode.");
    getch();
    closegraph();
}
```

Output:



Practical – 2(A)

Aim: To draw circle, rectangle, ellipse, sector and polygon on a screen.

Source Code:

```
#include<graphics.h>

#include<conio.h>

void main()

{

int gd=DETECT,gm;

int poly[12]={350,450,350,410,430,400,350,350,300,430,350,450};

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

circle(100,100,50);

outtextxy(75,170,"Circle");

rectangle(200,50,350,150);

outtextxy(240,170,"Rectangle");

ellipse(500,100,0,360,100,50);

outtextxy(480,170,"Ellipse");

line(100,250,540,250);

outtextxy(300,260,"Line");

sector(150,400,30,300,100,50);

outtextxy(120,460,"Sector");

drawpoly(6,poly);

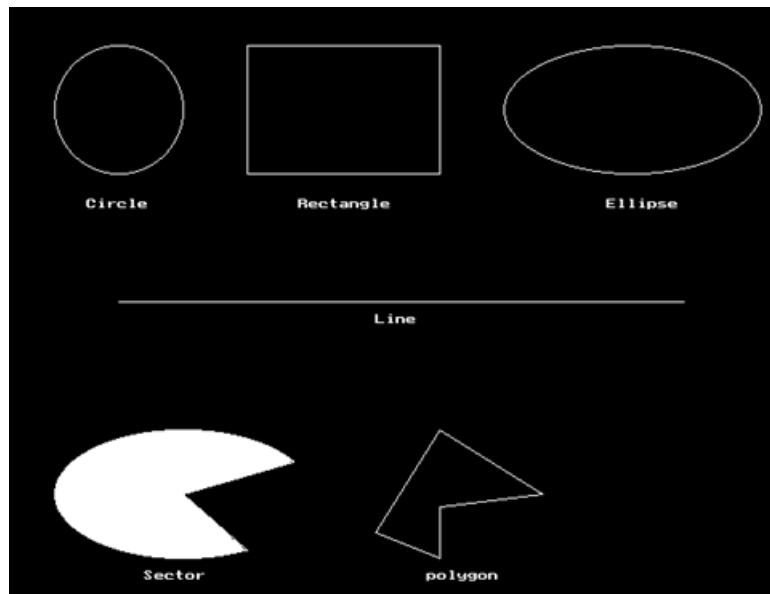
outtextxy(340,460,"polygon");

getch();

closegraph();

}
```


Output:



Practical – 2(B)

Aim: Draw a simple hut on the screen.

Source Code:

```
#include<graphics.h>

#include<conio.h>

void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    setcolor(WHITE);
    rectangle(150,180,250,300);
    rectangle(250,180,420,300);
    rectangle(180,250,220,300);
    line(200,100,150,180);
    line(200,100,250,180);
    line(200,100,150,100);
    line(370,100,420,180);
    setfillstyle(SOLID_FILL,BROWN);
    floodfill(152,182,WHITE);
    floodfill(252,182,WHITE);
    setfillstyle(SLASH_FILL,BLUE);
    floodfill(182,252,WHITE);
    setfillstyle(HATCH_FILL,GREEN);
    floodfill(200,105,WHITE);
    floodfill(210,105,WHITE);
    getch();
}
```

```
closegraph();  
}
```

Output:



Practical – 3

Aim: Draw the following basic shapes in the centre of the screen:

i. Circle ii. Rectangle iii. Square iv. Concentric Circles v. Ellipse vi. Line

Circle in Centre of the Screen:

```
#include<graphics.h>
#include<conio.h>
void main()
{
    int gd=DETECT,gm;
    int x,y,radius=80;
    initgraph(&gd,&gm,"C:\\\\TC\\\\BGI");
    x=getmaxx()/2;
    y=getmaxy()/2;
    outtextxy(x-100,50,"Circle Using Graphics in C");
    circle(x,y,radius);
    getch();
    closegraph();
}
```

Output:



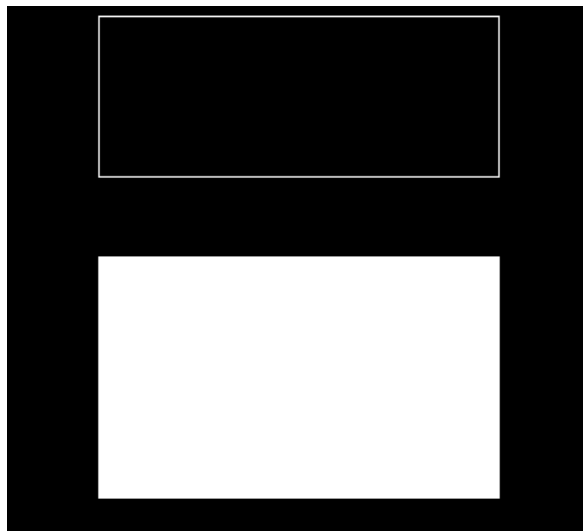
Rectangle in Centre of the Screen:

```
#include<graphics.h>

#include<conio.h>

void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    rectangle(150,50,400,150);
    bar(150,200,400,350);
    getch();
    closegraph();
}
```

Output:

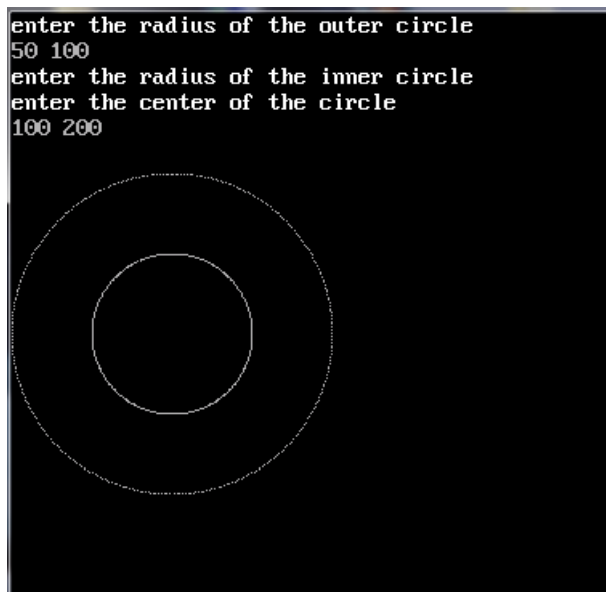


Concentric Circle in Centre of the Screen:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
int rc,rb,xc,yc,i;
float x,y;
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TC\\BGI");
printf("enter the radius of the outer circle\n");
scanf("%d",&rc);
printf("enter the radius of the inner circle\n");
scanf("%d",&rb);
printf("enter the center of the circle\n");
scanf("%d",&xc);
scanf("%d",&yc);
for(i=1;i<=360;i++)
{
x=xc+(rb*(cos (i)));
y=yc+(rb*(sin (i)));
putpixel(x,y,7);
}
for(i=1;i<=360;i++)
{
```

```
x=xc+(rc*(cos(i)));  
y=yc+(rc*(sin(i)));  
putpixel(x,y,7);  
}  
getch();  
closegraph();  
}
```

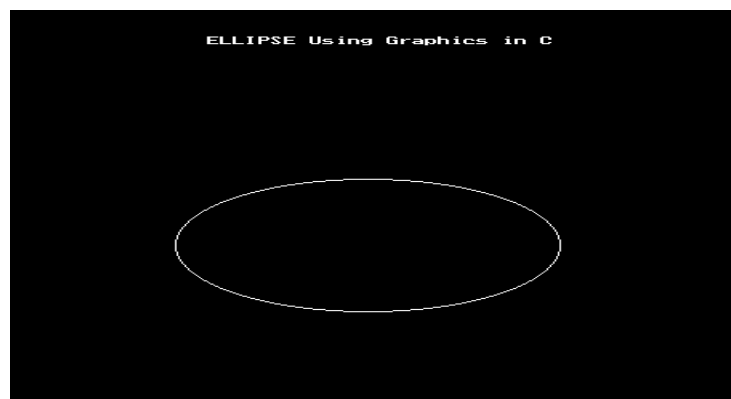
Output:



Ellipse Circle in Centre of the Screen:

```
#include<graphics.h>
#include<conio.h>
void main()
{
int gd=DETECT,gm;
int x,y;
initgraph(&gd,&gm,"C:\\TC\\BGI");
x=getmaxx()/2;
y=getmaxy()/2;
outtextxy(x-100,50,"ELLIPSE Using Graphics in C");
ellipse(x,y,0,360,120,60);
getch();
closegraph();
}
```

Output:



Line Circle in Centre of the Screen:

```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>

void main()

{

int gd=DETECT,gm;

int x1=200,y1=200;

int x2=300,y2=300;

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

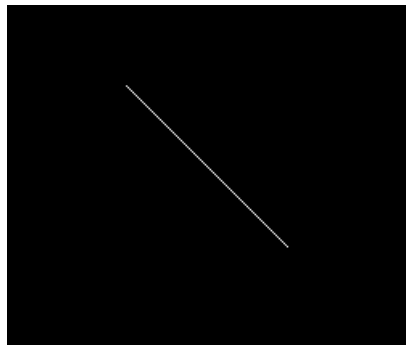
line(x1,y1,x2,y2);

getch();

closegraph();

}
```

Output:



Practical – 4(A)

Aim: Develop the program for DDA Line drawing algorithm.

Source Code:

```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<dos.h>

void main()

{

float x,y,x1,y1,x2,y2,dx,dy,step;

int i,gd=DETECT,gm;

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

printf("enter the value of x1 and y1");

scanf("%f%f",&x1,&y1);

printf("enter the value of x2 and y2");

scanf("%f%f",&x2,&y2);

dx=abs(x2-x1);

dy=abs(y2-y1);

if(dx>=dy)

step=dx;

else

step=dy;

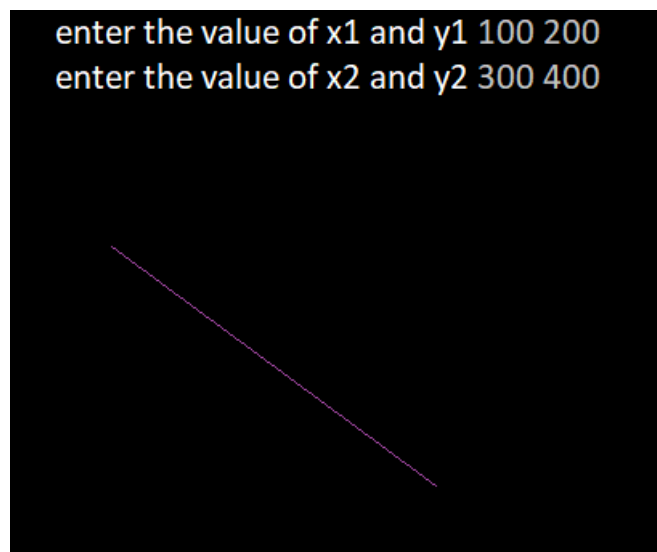
dx=dx/step;

dy=dy/step;

x=x1;
```

```
y=y1;  
i=1;  
while(i<=step)  
{  
    putpixel(x,y,5);  
    x=x+dx;  
    y=y+dy;  
    i=i+1;  
    delay(100);  
}  
closegraph();  
getch();  
}
```

Output:



Practical – 4(B)

Aim: Develop the program for Bresenham's Line drawing algorithm.

Source Code:

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void drawline(int x0,int y0,int x1,int y1)
{
    int dx,dy,p,x,y;

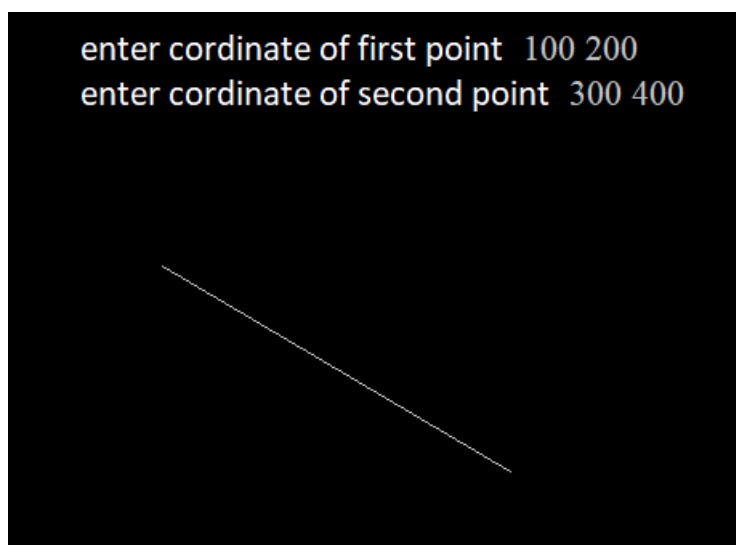
    dx=x1-x0;
    dy=y1-y0;
    x=x0;
    y=y0;
    p=2*dy-dx;
    while(x<x1)
    {
        if(p>=0)
        {
            putpixel(x,y,7);
            y=y+1;
            p=p+2*dy-2*dx;
        }
        else
        {
            putpixel(x,y,7);
            p=p+2*dy;
```

```

}
x=x+1;
}
}
void main()
{
int gd=DETECT,gm,error,x0,y0,x1,y1;
initgraph(&gd,&gm,"C:\\TC\\BGI");
printf("enter cordinate of first point");
scanf("%d%d",&x0,&y0);
printf("enter cordinate of second point");
scanf("%d%d",&x1,&y1);
drawline(x0,y0,x1,y1);
getch();
closegraph();
}

```

Output:



Practical – 5 (A)

Aim: Develop the program for the mid-point circle drawing algorithm.

Source Code:

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.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)

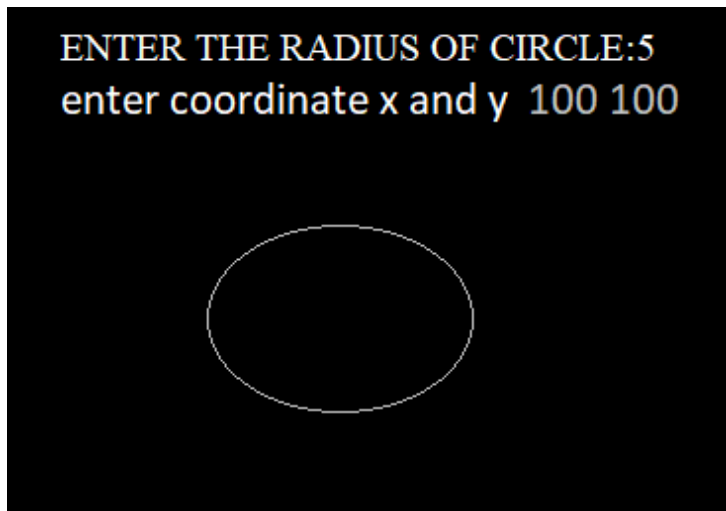
        {

            y+=1;

            err+=2*y+1;
```

```
}  
if(err>=0)  
{  
x-=1;  
err-=2*x+1;  
}  
}  
}  
void main()  
{  
int gd=DETECT,gm,error,x,y,r;  
initgraph(&gd,&gm,"C:\\TC\\BGI");  
printf("ENTER THE RADIUS OF CIRCLE:");  
printf("enter coordinate x and y");  
scanf("%d",&r);  
scanf("%d%d",&x,&y);  
drawcircle(x,y,r);  
closegraph();  
getch();  
}
```

Output:



Practical – 6 (A)

Aim: Write a program to perform 2D translation.

Source Code:

```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>

void main()

{

int graphdriver=DETECT,graphicmode;

int x1,y1,x2,y2,x,y,x3,y3,x4,y4;

printf("Enter the 2 lines end points:") ;

printf("x1,y1,x2,y2");

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

initgraph(&graphdriver,&graphicmode,"C:\\\\TC\\\\BGI");

line(x1,y1,x2,y2) ;

printf("enter the scaling co-ordinates:");

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

x3=x1+x;

y3=y1+y;

x4=x2+x;

y4=y2+y;

printf("Line after scaling");

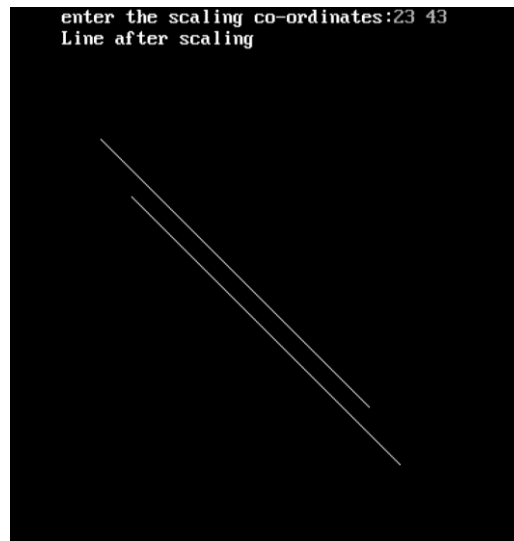
line(x3,y3,x4,y4) ;

getch();

closegraph();

}
```

Output:



Practical – 6 (B)

Aim: Write a program to implement 2D scaling.

Source Code:

```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>

void main()

{

int graphdriver=DETECT,graphicmode;

int x1,y1,x2,y2,x,y,x3,y3,x4,y4;

printf("Enter the 2 lines end points:") ;

printf("x1,y1,x2,y2");

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

initgraph(&graphdriver,&graphicmode,"C:\\TC\\BGI");

line(x1,y1,x2,y2) ;

printf("enter the scaling co-ordinates:");

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

x3=x1*x;

y3=y1*y;

//x4=x2*x;

//y4=y2*y;

printf("Line after scaling");

line(x1,y1,x4,y4) ;

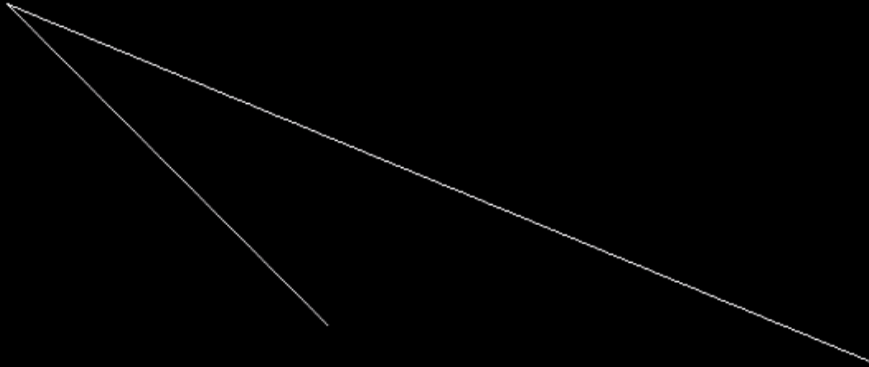
getch();

closegraph();

}
```

Output:

```
enter the scaling co-ordinates:4  
5  
Line after scaling
```



Practical – 6 (C) 7 (A)

Aim: Perform 2D Rotation on a given object.

Source Code:

```
#include<graphics.h>

#include<stdlib.h>

#include<stdio.h>

#include<math.h>

#include<conio.h>

void main()

{

int graphdriver = DETECT,graphmode;

int i;

int x2,y2,x1,y1,x,y,xn,yn;

double r11,r12,r21,r22,th;

clrscr();

printf("Enter the 2 lie end points:");

printf("x1,y1,x2,y2");

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

initgraph(&graphdriver,&graphmode,"C:\\TC\\BGI");

line(x1,y1,x2,y2);

printf("\n\n\nEnter the angle");

scanf("%lf",&th);

r11=cos((th*3.1428)/180);

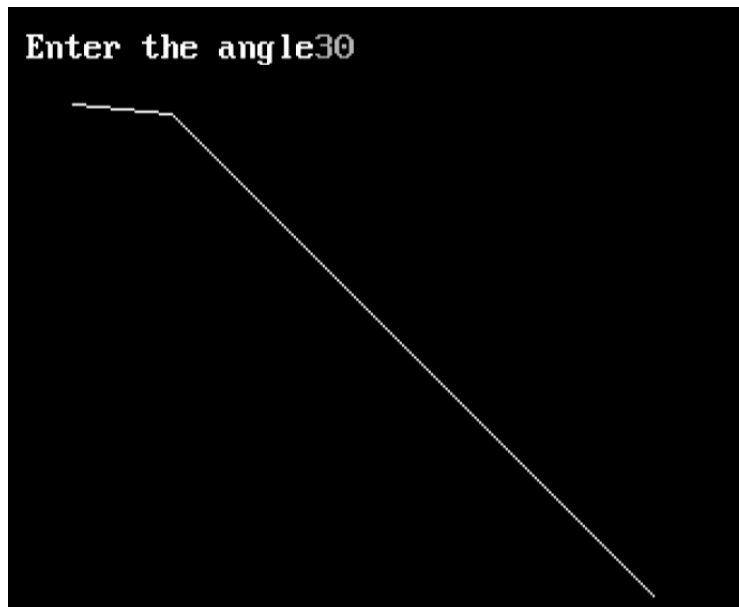
r12=sin((th*3.1428)/180);

r21=(-sin((th*3.1428)/180));

r22=cos((th*3.1428)/180);
```

```
xn=((x2*r11)-(y2*r12));  
yn=((x2*r21)+(y2*r22));  
line(x1,y1,xn,yn);  
getch();  
}
```

Output:



Practical – ~~7~~(A) 9(A)

Aim: Write a program to fill a circle using Flood Fill Algorithm.

Source Code:

```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<dos.h>

void floodfill(int x,int y,int oldcolor,int newcolor)

{

if(getpixel(x,y)==oldcolor)

{

delay(20);

putpixel(x,y,newcolor);

floodfill(x+1,y,oldcolor,newcolor);

floodfill(x,y+1,oldcolor,newcolor);

floodfill(x-1,y,oldcolor,newcolor);

floodfill(x,y-1,oldcolor,newcolor);

}

}

void main()

{

int gd=DETECT,gm,radius;

int x, y;

printf("enter x and y position for circle\n");

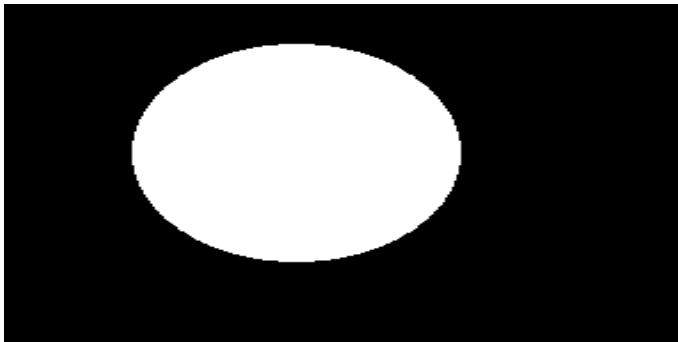
scanf("%d%d",&x,&y);

printf("enter radius of circle\n");
```

```
scanf("%d",&radius);  
initgraph(&gd,&gm,"c:\\TC\\BGI");  
circle(x,y,radius);  
floodfill(x,y,0,15);  
closegraph();  
getch();  
}
```

Output:

```
enter x and y position for circle  
100 200  
enter radius of circle  
30
```



Practical – ~~7(B)~~ 9(B)

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

Source Code:

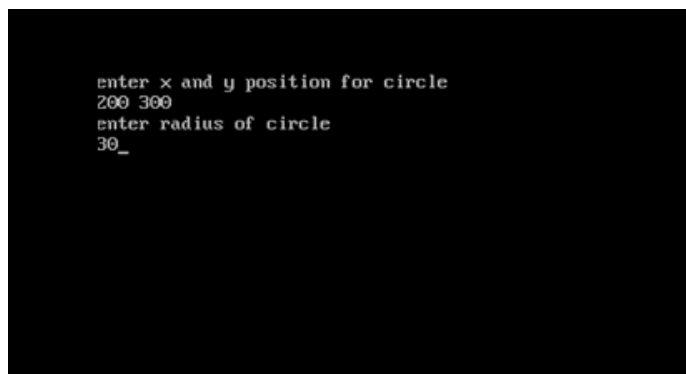
```
#include<graphics.h>
#include<iostream.h>
#include<dos.h>
#include<conio.h>

void boundaryfill(int x ,int y,int f_color,int b_color)
{
    if(getpixel(x,y)!=b_color && getpixel(x,y)!=f_color)
    {
        delay(20);
        putpixel(x,y,f_color);
        boundaryfill(x+1,y,f_color,b_color);
        boundaryfill(x,y+1,f_color,b_color);
        boundaryfill(x-1,y,f_color,b_color);
        boundaryfill(x,y-1,f_color,b_color);
    }
}

int main()
{
    int gd=DETECT,gm,radius;
    int x,y;
    cout<<"enter x and y position for circle\n";
    cin>>x>>y;
```

```
cout<<"enter radius of circle\n";  
cin>>radius;  
initgraph(&gd,&gm,"c:\\TC\\BGI");  
circle(x,y,radius);  
boundaryfill(x,y,4,15);  
closegraph();  
return 0;  
}
```

Output:



```
enter x and y position for circle  
200 300  
enter radius of circle  
30_
```



Practical – 8 (A)

Aim: Develop a simple text screen saver using graphics functions.

Source Code:

```
#include<graphics.h>

#include<conio.h>

#include<stdio.h>

#include<dos.h>

void main()

{

int gd=DETECT,gm,i,maxx,maxy,key0;

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

maxx=getmaxx();

maxy=getmaxy();

while(!kbhit())

{

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

{

cleardevice();

settextstyle(2,0,5);

outtextxy(maxx/2,i,"C Graphics");

delay(100);

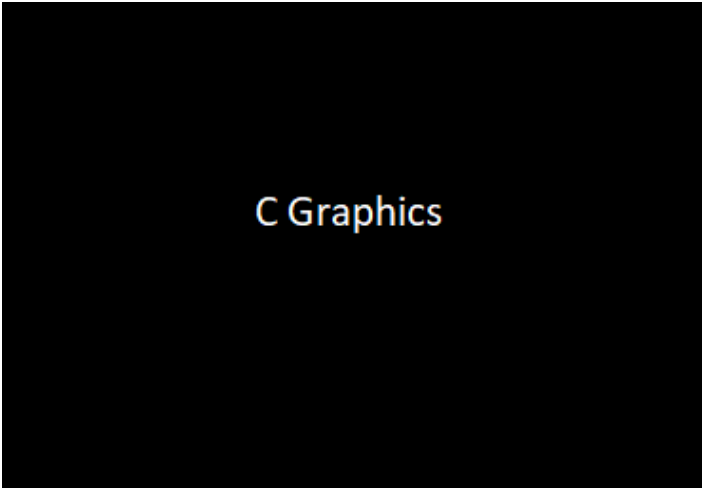
}

}

getch();

}
```

Output:



C Graphics

Practical – 8 (B)

Aim: Perform smiling face animation using graphic functions.

Source Code:

```
#include<graphics.h>

#include<conio.h>

#include<stdio.h>

void main()

{

int gd=DETECT,gm;

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

circle(200,200,30);

circle(190,190,5);

arc(190,190,50,130,10);

circle(210,190,5);

arc(210,190,50,130,10);

arc(200,210,180,360,10);

line(187,210,193,210);

line(207,210,213,210);

line(198,195,195,200);

line(202,195,205,200);

line(195,200,200,205);

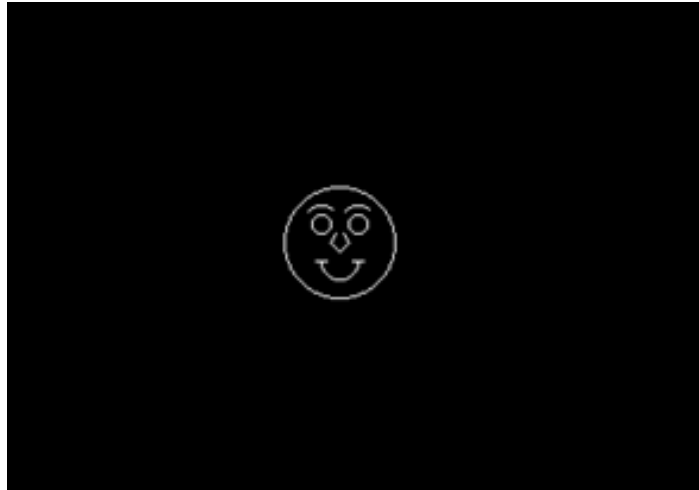
line(205,200,200,205);

getch();

closegraph();

}
```

Output:



Practical – 8 (C)

Aim: Draw the moving car on the screen.

Source Code:

```
#include<graphics.h>

#include<dos.h>

#include<conio.h>

void main()

{

int i,j=0,gd=DETECT,gm;

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

settextstyle(DEFAULT_FONT,HORIZ_DIR,2);

outtextxy(25,240,"press any key to view the moving car");

getch();

setviewport(0,0,639,440,1);

for(i=0;i<=420;i=i+10,j++)

{

rectangle(50+i,275,150+i,400);

rectangle(150+i,350,200+i,400);

circle(75+i,410,10);

circle(175+i,410,10);

setcolor(j);

delay(100);

if(i==420);

break;

clearviewport();

}
```



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

Output:

press any key to view the moving car

