LAB PRACTICALS

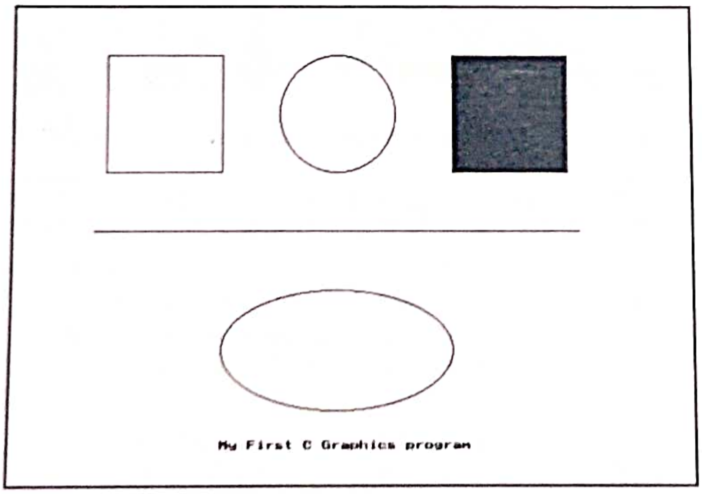
PRACTICAL NO : 1A

- **AIM :** Study and enlist the basic functions used for graphics in C / C++ language. Give an example for each of them.

☞ PROGRAM

```
#include<graphics.h>
#include<conio.h>
main()
{
int
gd=DETECT,gm,left=100,top=100,right=200,bottom=200,
x=300,y=150,radius=50;
initgraph(&gd,&gm,"c:\\TURBOC3\\BGI");
rectangle(left,top,right,bottom);
circle(x,y,radius);
bar(left+300,top,right+300,bottom);
line(left-10,top+150,left+410,top+150);
ellipse(x,y+200,0,360,100,50);
outtextxy(left+100,top+325,"My First C Graphics program");
getch();
closegraph();
return 0;
}
```

☞ Output



```
My First C Graphics program
```

☞ Functions

- **Arc function in c**
  Declaration : void arc(int x, int y, int stangle, int endangle, int radius);
- **Circle function in c**
  Declaration : void circle(int x, int y, int radius);
- **closegraph function in c**
  Declaration : void closegraph();
- **Floodfill function**
  Declaration : void floodfill(int x, int y, int border);
- **getx function in C**
  Declaration : int getx();
- **gety function in c**
  Declaration : int gety();
- **putpixel function in c**
  Declaration : void putpixel(int x, int y, int color);
- **outtext function**
  Declaration : void outtext(char *string);

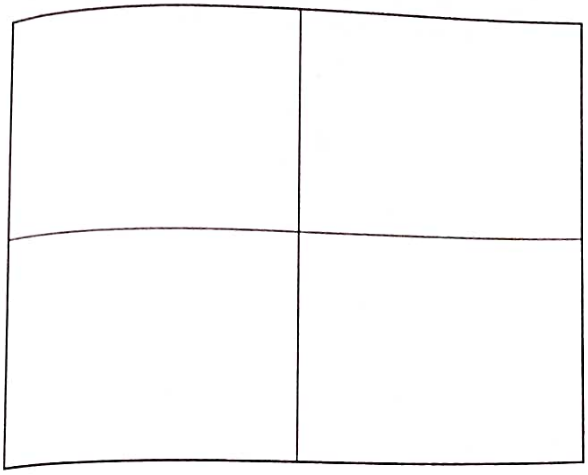
PRACTICAL NO : 1B

- **AIM :** Draw a co-ordinate axis at the center of the screen.

☞ PROGRAM

```
#include <graphics.h>
#include <conio.h>
main()
{
    int gd=DETECT,gm;
    int midx,midy;
    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
    midx=getmaxx()/2;
    midy=getmaxy()/2;
    line(1,midy,640,midy);
    line(midx,1,midx,640);
    getch();
    closegraph();
    return 0;
}
```

☞ **Output**



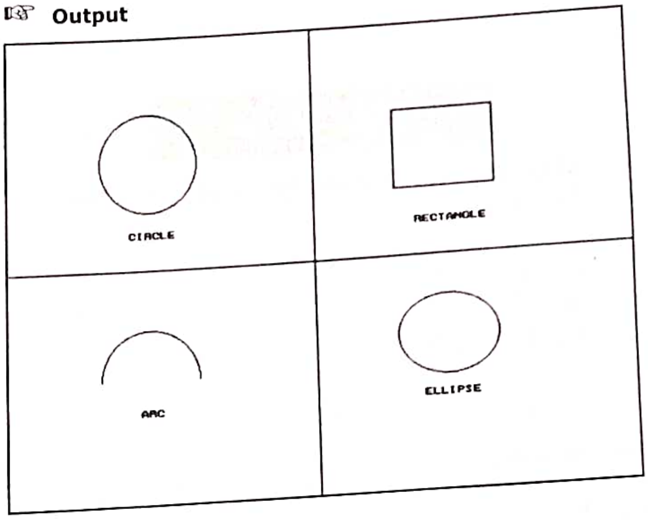
PRACTICAL NO : 2A

- **AIM :** Divide your screen into four region, draw circle, rectangle, ellipse and half ellipse in each region with appropriate message.

☞ **PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
main()
{
int gd=DETECT,gm;
int midx,midy;
initgraph(&gd,&gm,"c:\\TURBOC3\\bgi");
midx=getmaxx()/2;
midy=getmaxy()/2;
line(1,midy,840,midy);
line(midx,1,midx,940);
circle(150,130,50);
outtextxy(130,200,"CIRCLE");
rectangle(400,90,500,170);
outtextxy(420,200,"RECTANGLE");
arc(150,350,0,180,50);
outtextxy(140,380,"ARC");
ellipse(450,320,0,360,50,40);
outtextxy(425,375,"ELLIPSE");
getch();
closegraph();
return 0;
}
```

☞ **Output**



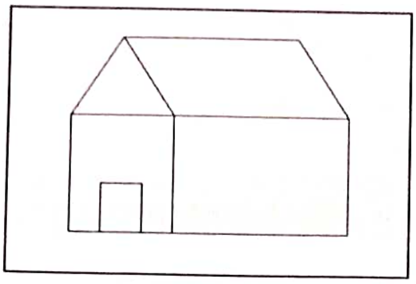
PRACTICAL NO : 2B

- **AIM :** Draw a simple hut on the screen.

☞ **PROGRAM**

```
#include<graphics.h>
#include<conio.h>
int main(){
int gd = DETECT,gm;
 initgraph(&gd, &gm, "c:\\TURBOC3\\BGI");
 /* Draw Hut */
 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(200,100,370,100);
 line(370,100,420,180);
 getch();
 closegraph();
 return 0;
 }
```

☞ **Output**



PRACTICAL NO : 3

- **AIM :** Draw the following basic shapes in the center of the screen :
  - i . Circle
  - ii. Rectangle
  - iii. Square
  - iv. Concentric Circles
  - v. Ellipse
  - vi. Line

☞ **PROGRAM**

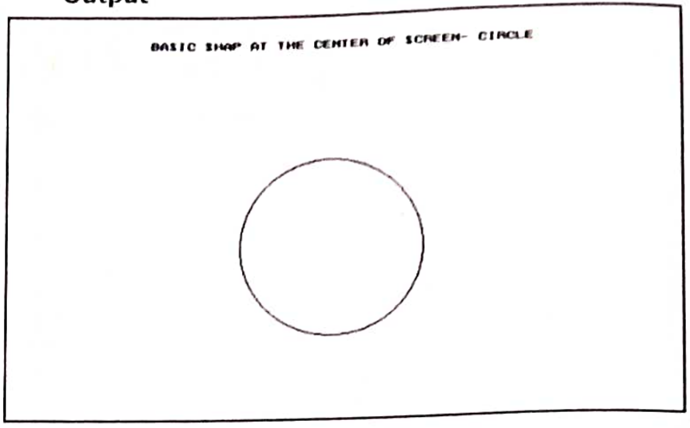
i . Circle

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main(){
 int gd = DETECT,gm;
 int x ,y ,radius=80;
 initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
 x = getmaxx()/2;
 y = getmaxy()/2;

 outtextxy(160,50, "BASIC SHAPE AT THE CENTER OF
SCREEN- CIRCLE");
 circle(x, y, radius);
 getch();
 closegraph();
 return 0;

}
```

☞ **Output**



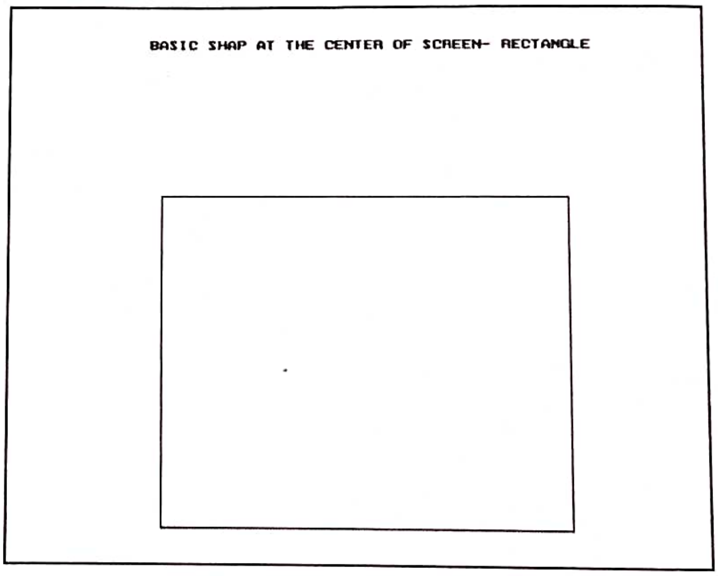
```
BASIC SHAP AT THE CENTER OF SCREEN- CIRCLE
```

ii. Rectangle

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main(){
 int gd = DETECT,gm;
 int x ,y;
 initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

 outtextxy(160,50, "BASIC SHAPE AT THE CENTER OF
SCREEN- RECTANGLE");
 rectangle(170,420,500,170);
 getch();
 closegraph();
 return 0;
}
```

☞ **Output**

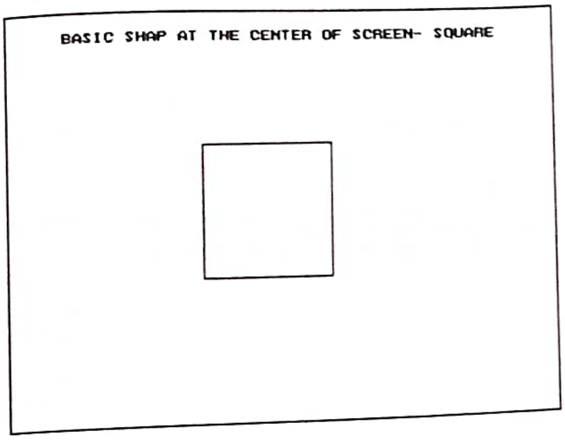


```
BASIC SHAP AT THE CENTER OF SCREEN- RECTANGLE
```

iii. Square

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main(){
 int gd = DETECT,gm;
 int x ,y ;
 initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
 outtextxy(160,50, "BASIC SHAPE AT THE CENTER OF
SCREEN- SQUARE");
 rectangle(250,180,380,340);
 getch();
 closegraph();
 return 0;
}
```

Output

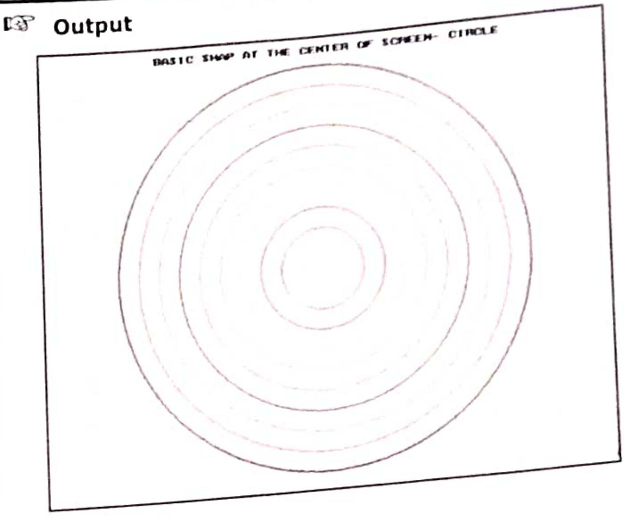


```
BASIC SHAP AT THE CENTER OF SCREEN- SQUARE
```

iv. Concentric Circles

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main(){
 int gd = DETECT,gm,color=1;
 int x ,y,i;
 initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
 x = getmaxx()/2;
 y = getmaxy()/2;
 outtextxy(160,20, "BASIC SHAPE AT THE CENTER OF
SCREEN- CIRCLE");
 for(i=20;i<=200;i+=20){
 setcolor(color++);
 circle(x,y,i); }
 getch();
 closegraph();
 return 0;
}
```

Output

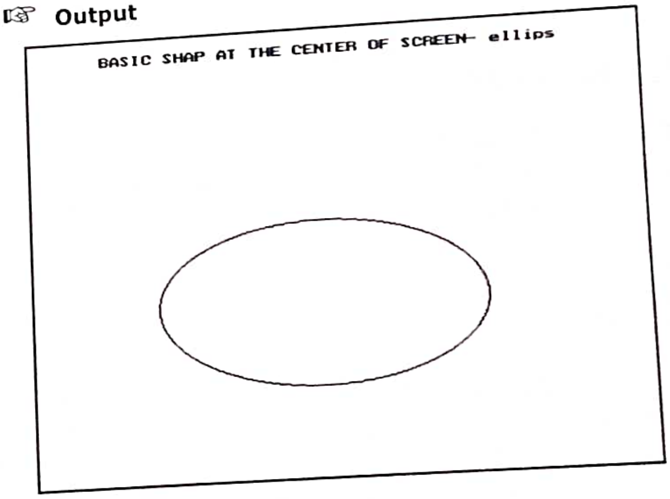


```
BASIC SHAP AT THE CENTER OF SCREEN- CIRCLE
```

v. Ellipse

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main(){
 int gd = DETECT,gm;
 int x ,y;
 initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
 x = getmaxx()/2;
 y = getmaxy()/2;
 outtextxy(160,50, "BASIC SHAPE AT THE CENTER OF
SCREEN- ellipse");
 ellipse(x, y, 0, 360, 120, 60);
 getch();
 closegraph();
 return 0;
}
```

Output

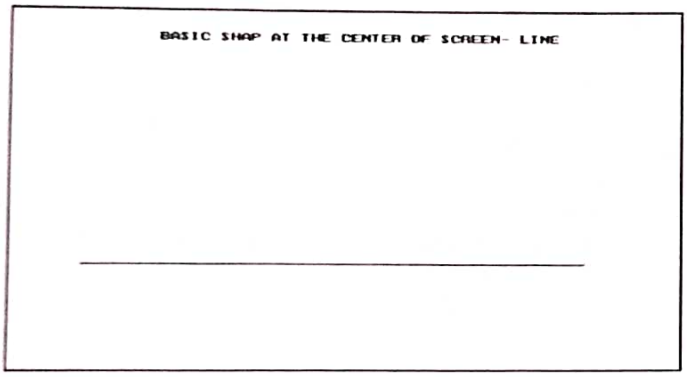


```
BASIC SHAP AT THE CENTER OF SCREEN- ellips
```

vi. Line

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main(){
 int gd = DETECT,gm;
 int x ,y ;
 initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
 x = getmaxx()/2;
 y = getmaxy()/2;
 outtextxy(160,50, "BASIC SHAPE AT THE CENTER OF
SCREEN- LINE");
 line(100,250,500,250);
 getch();
 closegraph();
 return 0;
}
```

Output



```
BASIC SHAP AT THE CENTER OF SCREEN- LINE
```

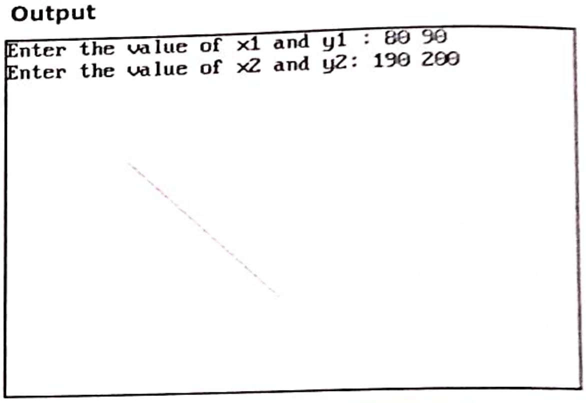
PRACTICAL NO : 4A

- **AIM :** Develop the program for DDA Line drawing algorithm.

PROGRAM

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
void main( )
{
float x,y,x1,y1,x2,y2,dx,dy,step;
int i,gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TURBOC3\\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();
}
```

Output



```
Enter the value of x1 and y1 : 80 90
Enter the value of x2 and y2: 190 200
```

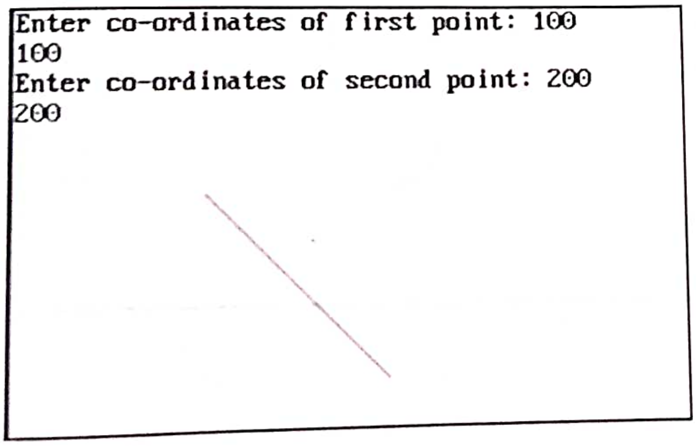
PRACTICAL NO : 4B

- **AIM :** Develop the program for Bresenham's Line drawing algorithm.

PROGRAM

```
#include<math.h>
#include<graphics.h>
#include<conio.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;
}
}
int main()
{
int gdriver=DETECT, gmode, error, x0, y0, x1, y1;
clrscr();
initgraph(&gdriver, &gmode, "c:\\TURBOC3\\bgi");
printf("Enter coordinates of first point: ");
scanf("%f%f",&x0,&y0);
printf("Enter coordinates of second point: ");
scanf("%f%f",&x1,&y1);
drawline(x0, y0, x1, y1);
closegraph();
return 0;
}
```

☞ **Output**



```
Enter co-ordinates of first point: 100
100
Enter co-ordinates of second point: 200
200
```

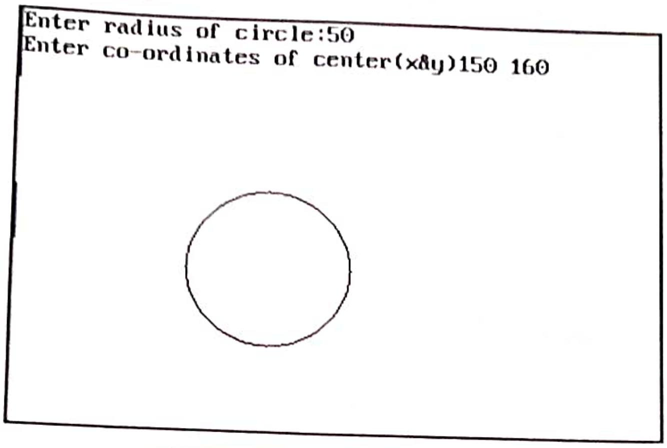
PRACTICAL NO : 5A

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

☞ **PROGRAM**

```
#include<iostream.h>
#include<graphics.h>
#include<conio.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;
}
}
}
int main()
{
int gddriver=DETECT,gmode,error,x,y,r;
initgraph(&gddriver,&gmode,"C:\\TURBOC3\\BGI");
cout<<"Enter radius of circle:";
cin>>r;
cout<<"Enter co-ordinates of center(x&y)";
cin>>x>>y;
drawcircle(x,y,r);
getch();
return 0;
}
```

Output



```
Enter radius of circle:50
Enter co-ordinates of center(x&y)150 160
```

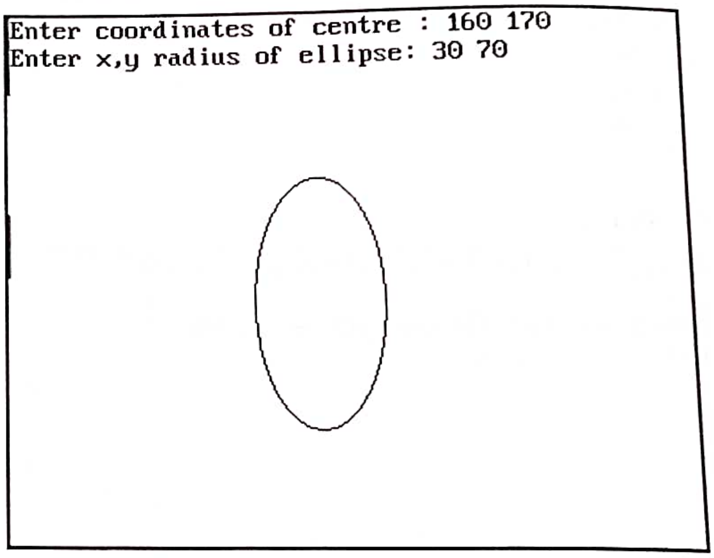
PRACTICAL NO : 5B

- **AIM :** Develop the program for the mid-point ellipse drawing algorithm.

PROGRAM

```
#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:\\TURBOC3\\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



```
Enter coordinates of centre : 160 170
Enter x,y radius of ellipse: 30 70
```

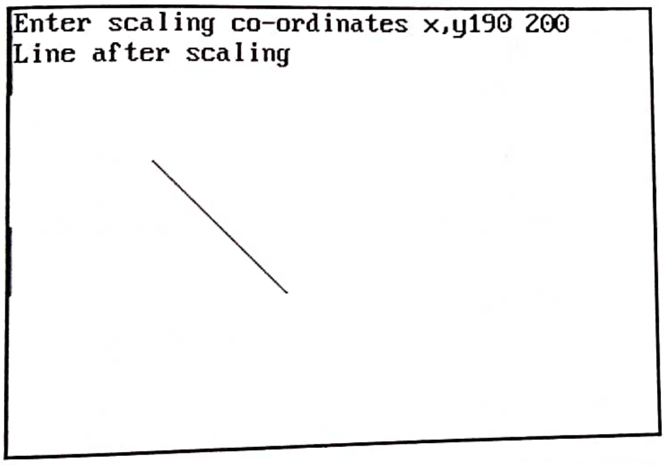
PRACTICAL NO : 6A

- **AIM :** Write a program to implement 2D scaling.

PROGRAM

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
void main()
{
int graphdriver=DETECT,graphmode,errorcode;
int i;
int x2,y2,x1,y1,x,y;
printf("Enter the 2 line end points:");
printf("x1,y1,x2,y2");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
initgraph(&graphdriver,&graphmode,"C:\\TURBOC3\\BGI");
line(x1,y1,x2,y2);
printf("Enter scaling co-ordinates ");
printf("x,y");
scanf("%d%d",&x,&y);
x1=(x1*x);
y1=(y1*y);
x2=(x2*x);
y2=(y2*y);
printf("Line after scaling");
line(x1,y1,x2,y2);
getch();
closegraph();
}
```

Output



```
Enter scaling co-ordinates x,y190 200
Line after scaling
```

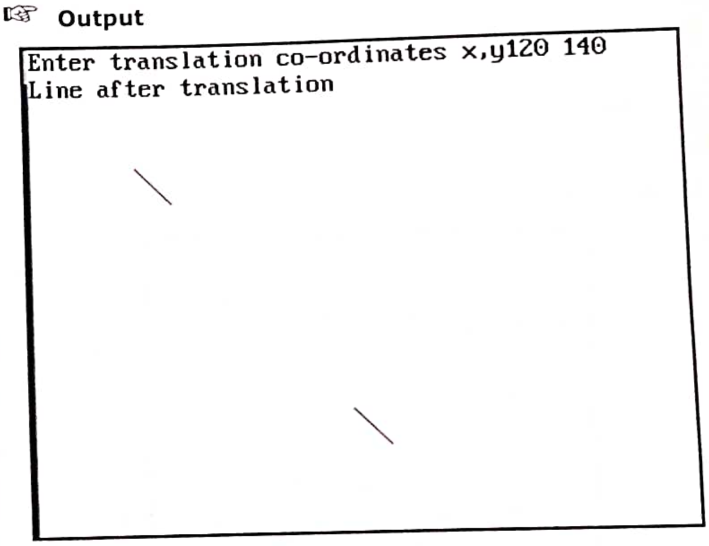
PRACTICAL NO : 6B

- **AIM :** Write a program to perform 2D translation.

PROGRAM

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
void main()
{
int graphdriver=DETECT,graphmode,errorcode;
int i;
int x2,y2,x1,y1,x,y;
printf("Enter the 2 line end points:");
printf("x1,y1,x2,y2");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
initgraph(&graphdriver,&graphmode,"C:\\TURBOC3\\BGI");
line(x1,y1,x2,y2);
printf("Enter translation co-ordinates ");
printf("x,y");
scanf("%d%d",&x,&y);
x1=x1+x;
y1=y1+y;
x2=x2+x;
y2=y2+y;
printf("Line after translation");
line(x1,y1,x2,y2);
getch();
closegraph();
}
```

Output



```
Enter translation co-ordinates x,y120 140
Line after translation
```

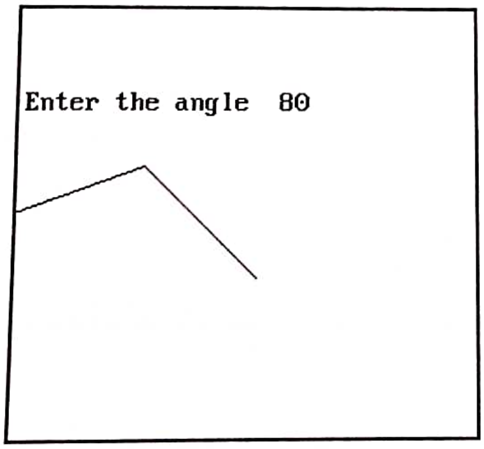
PRACTICAL NO : 7A

- **AIM :** Perform 2D Rotation on a given object.

☞ PROGRAM

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include<conio.h>
void main()
{
int graphdriver=DETECT,graphmode,errorcode;
int i;
int x2,y2,x1,y1,x,y,xn,yn;
double r11,r12,th;
float r21,r22;
clrscr();
printf("Enter the 2 line end points:");
printf("x1,y1,x2,y2 ");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
initgraph(&graphdriver,&graphmode,"C:\\TURBOC3\\BGI");
line(x1,y1,x2,y2);
printf("\n\n\n  Enter 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);
//printf("%lf %lf %lf %lf",r11,r12,r21,r22);
xn=((x2*r11)-(y2*r12));
yn=((x2*r12)+(y2*r11));
line(x1,y1,xn,yn);
getch();
closegraph();
}
```

☞ Output



```
Enter the angle  80
```

PRACTICAL NO : 7B

- **AIM :** Program to create a house like figure and perform the following operations.
  i. Scaling about the origin followed by translation.
  ii. Scaling with reference to an arbitrary point.
  iii. Reflect about the line y = mx + c.

☞ PROGRAM

```
#include <stdio.h>
#include <graphics.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
void reset (int h[][2])
{
int val[9][2] = {

{ 50, 50 },{ 75, 50 },{ 75, 75 },{ 100, 75 },
{ 100, 50 },{ 125, 50 },{ 125, 100 },{ 87, 125 },{ 50, 100 }
};


int i;
for (i=0; i<9; i++)
{
h[i][0] = val[i][0]-50;
h[i][1] = val[i][1]-50;
}
}
void draw (int h[][2])
{
int i;
setlinestyle (DOTTED_LINE, 0, 1);
line (320, 0, 320, 480);
line (0, 240, 640, 240);
setlinestyle (SOLID_LINE, 0, 1);
for (i=0; i<8; i++)
line (320+h[i][0], 240-h[i][1], 320+h[i+1][0], 240-
h[i+1][1]);
line (320+h[0][0], 240-h[0][1], 320+h[8][0], 240-h[8][1]);
}
void rotate (int h[][2], float angle)
{
int i;
for (i=0; i<9; i++)
{
```

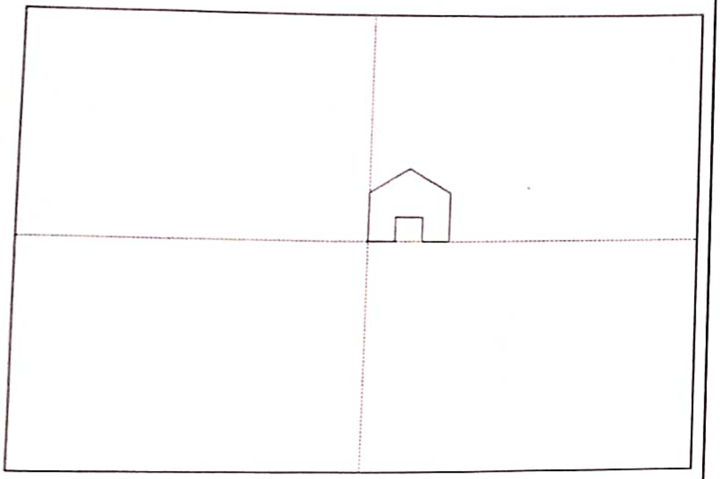
```
int xnew, ynew;
xnew = h[i][0] * cos (angle) - h[i][1] * sin (angle);
ynew = h[i][0] * sin (angle) + h[i][1] * cos (angle);
h[i][0] = xnew; h[i][1] = ynew;
}
}
void scale (int h[][2], int sx, int sy)
{
int i;
for (i=0; i<9; i++)
{
h[i][0] *= sx;
h[i][1] *= sy;
}
}
void translate (int h[][2], int dx, int dy)
{
int i;
for (i=0; i<9; i++)
{
h[i][0] += dx;
h[i][1] += dy;
}
}
void reflect (int h[][2], int m, int c)
{
int i;
float angle;
for (i=0; i<9; i++)
h[i][1] -= c;
angle = M_PI/2 - atan (m);
rotate (h, angle);
for (i=0; i<9; i++)
h[i][0] = -h[i][0];
angle = -angle;
rotate (h, angle);
for (i=0; i<9; i++)
h[i][1] += c;

}
void ini()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
}
void dini()
{
getch();
closegraph();
}
```

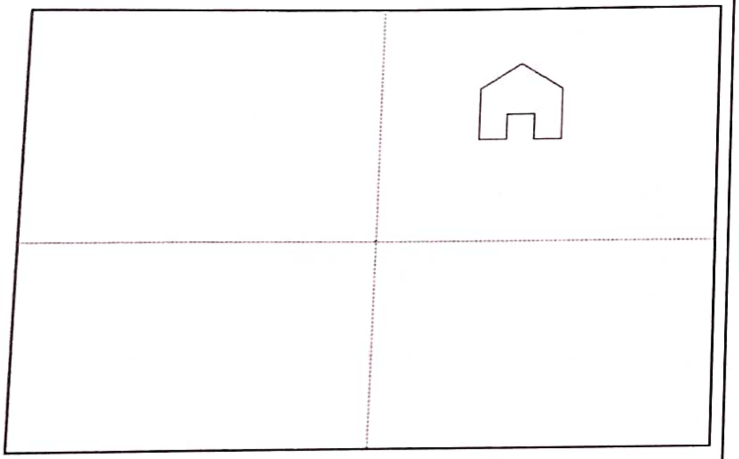
```
void main()
{
int h[9][2],sx,sy,x,y,m,c,choice;
do
{
clrscr();
printf("1. Scaling about the origin.\n");
printf("2. Scaling about an arbitrary point.\n");
printf("3. Reflection about the line y = mx + c.\n");
printf("4. Exit\n");
printf("Enter the choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1: printf ("Enter the x- and y-scaling factors: ");
scanf ("%d%d", &sx, &sy);
ini();
reset (h);
draw (h);getch();
scale (h, sx, sy);
cleardevice();
draw (h);
dini();
break;
case 2: printf ("Enter the x- and y-scaling factors: ");
scanf ("%d%d", &sx, &sy);
printf ("Enter the x- and y-coordinates of the point: ");
scanf ("%d%d", &x, &y);
ini();
reset (h);
translate (h, x, y);// Go to arbitrary point
draw(h); getch();//Show its arbitrary position
cleardevice();
translate(h,-x,-y);//Take it back to origin
draw(h);
getch();
cleardevice();
scale (h, sx, sy);//Now Scale it
draw(h);
getch();
translate (h, x, y);//Back to Arbitrary point
cleardevice();
draw (h);
putpixel (320+x, 240-y, WHITE);
dini();
break;
case 3: printf ("Enter the values of m and c: ");
scanf ("%d%d", &m, &c);
ini();
reset (h);
draw (h); getch();
reflect (h, m, c);
cleardevice();
draw (h);
dini();
break;
case 4: exit(0);
}
}while(choice!=4);
}
```

Output

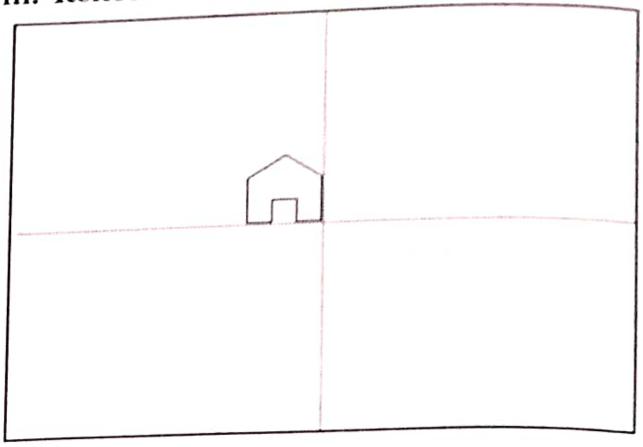
i. Scaling about the origin followed by translation.



ii. Scaling with reference to an arbitrary point



iii. Reflect about the line y = mx + c.



PRACTICAL NO : 8A

- **AIM :** Write a program to implement Cohen-Sutherland clipping.

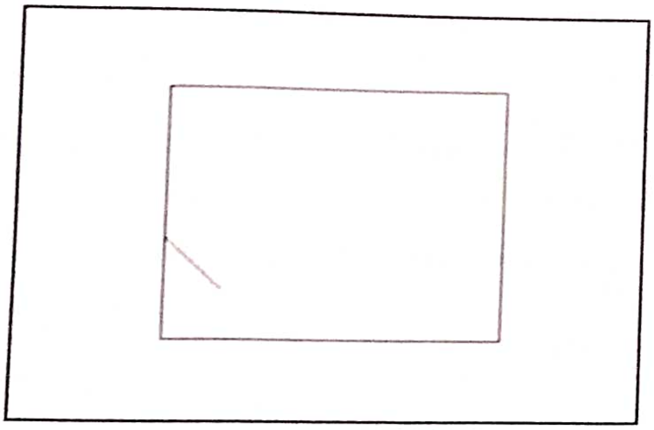
PROGRAM

```
#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(500);
drawline(p1,p2);
delay(500);
```

```
cleardevice();
delay(500);
p1=setcode(p1);
p2=setcode(p2);
v=visibility(p1,p2);
delay(500);
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) //for setting the 4 bit code
{
PT ptemp;
if(p.y<100)
ptemp.code[0]='1'; //Top
else
ptemp.code[0]='0';
if(p.y>350)
ptemp.code[1]='1'; //Bottom
else
ptemp.code[1]='0';
if(p.x>450)
ptemp.code[2]='1'; //Right
else
ptemp.code[2]='0';
if(p.x<150)
ptemp.code[3]='1'; //Left
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**



PRACTICAL NO : 8B

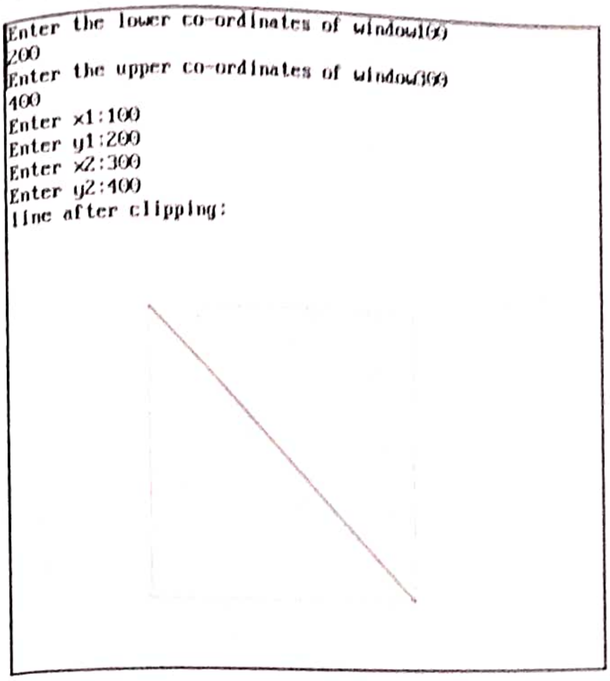
- **AIM :** Write a program to implement Liang - Barsky Line Clipping Algorithm

☞ **PROGRAM**

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
int
x1,y1,x2,y2,xmax,xmin,ymax,ymin,xx1,yy1,xx2,yy2,dx,dy,i;
int p[4],q[4];
float t1,t2,t[4];
cout<<"Enter the lower co-ordinates of window";
cin>>xmin>>ymin;
cout<<"Enter the upper co-ordinates of window";
cin>>xmax>>ymax;
setcolor(RED);
rectangle(xmin,ymin,xmax,ymax);
cout<<"Enter x1:";
cin>>x1;
cout<<"Enter y1:";
cin>>y1;
cout<<"Enter x2:";
cin>>x2;
cout<<"Enter y2:";
cin>>y2;
line(x1,y1,x2,y2);
dx=x2-x1;
dy=y2-y1;
p[0]=-dx;
p[1]=dx;
p[2]=-dy;
p[3]=dy;
q[0]=x1-xmin;
q[1]=xmax-x1;
q[2]=y1-ymin;
q[3]=ymax-y1;
for(i=0;i < 4;i++){
if(p[i]!=0){
t[i]=(float)q[i]/p[i];
}
else
if(p[i]==0 && q[i] < 0)
cout<<"line completely outside the window";
else
if(p[i]==0 && q[i] >= 0)
cout<<"line completely inside the window";

}
if (t[0] > t[2]){
t1=i[0];
}
else{
t1=t[2];
}
if (t[1] < t[3]){
t2=t[1];
}
else{
t2=t[3];
}
if (t1 < t2){
xx1=x1+t1*dx;
xx2=x1+t2*dx;
yy1=y1+t1*dy;
yy2=y1+t2*dy;
cout<<"line after clipping:";
setcolor(WHITE);
line(xx1,yy1,xx2,yy2);
}
else{
cout<<"line lies out of the window";
}
getch();
}
```

☞ **Output**



```
Enter the lower co-ordinates of window100
200
Enter the upper co-ordinates of window300
400
Enter x1:100
Enter y1:200
Enter x2:300
Enter y2:400
line after clipping:
```

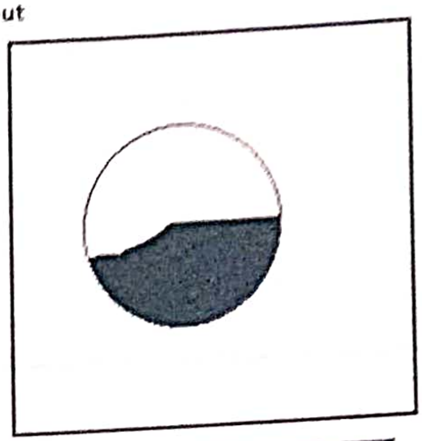
PRACTICAL NO : 9A

- **AIM :** Write a program to fill a circle using Flood Fill Algorithm.

☞ **PROGRAM**

```
#include<stdio.h>
#include<graphics.h>
#include<dos.h>
void floodFill(int x,int y,int oldcolor,int newcolor)
{
if(getpixel(x,y) == oldcolor)
{
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);
}
}
//getpixel(x,y) gives the color of specified pixel
int main()
{
int gm,gd=DETECT,radius;
int x,y;
printf("Enter x and y positions for circle\n");
scanf("%d%d",&x,&y);
printf("Enter radius of circle\n");
scanf("%d",&radius);
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
circle(x,y,radius);
floodFill(x,y,0,15);
delay(5000);
closegraph();
return 0;
}
```

☞ **Output**



PRACTICAL NO : 9B

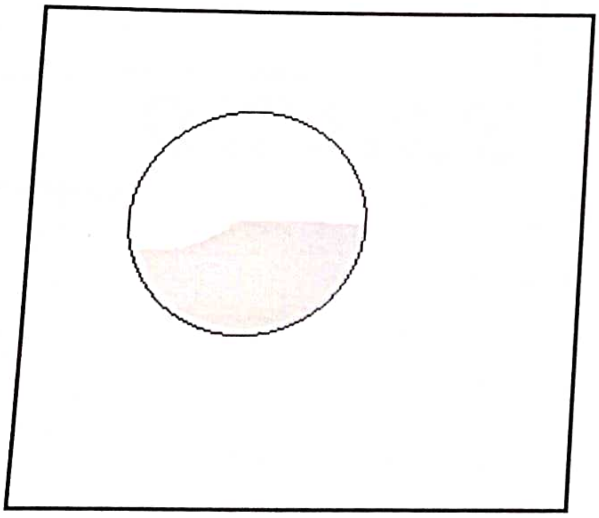
- **AIM :** Write a program to fill a circle using Boundary Fill Algorithm.

☞ **PROGRAM**

```
#include<iostream.h>
#include<graphics.h>
#include<dos.h>
void boundaryfill(int x,int y,int f_color,int b_color)
{
if(getpixel(x,y)!=b_color && getpixel(x,y)!=f_color)
{
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 gm,gd=DETECT,radius;
int x,y;
cout<<"Enter x & y positions for circle \n";
cin>>x>>y;
cout<<"Enter radius of circle \n";
cin>>radius;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
circle(x,y,radius);
boundaryfill(x,y,4,15);
delay(5000);
closegraph();
return 0;
}
```

☞ **Output**



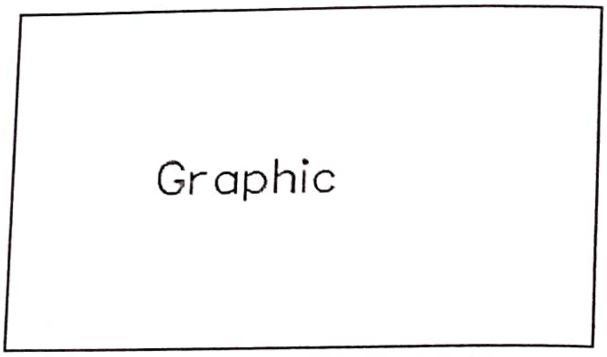
PRACTICAL NO : 10A

- **AIM :** Develop a simple text screen saver using graphics functions.

☞ **PROGRAM**

```
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm,maxx,maxy;
initgraph(&gd,&gm,"c:\\tc\\bgi");
maxx=getmaxx()/2;
maxy=getmaxy()/2;
while(!kbhit())
{
for(int i=0;i<maxy;i++)
{
cleardevice();
settextstyle(3,0,5);
outtextxy(maxx/2,i,"Graphics c");
} }
getch();
}
```

☞ **Output**



```
Graphic
```

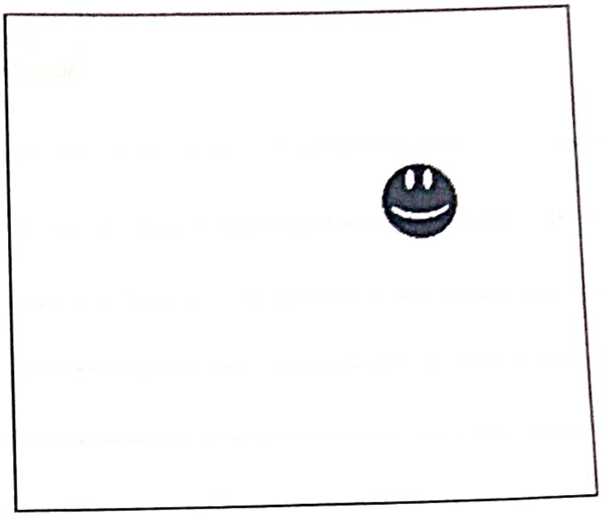
PRACTICAL NO : 10B

- **AIM :** Perform smiling face animation using graphic functions.

☞ **PROGRAM**

```
#include<graphics.h>
#include<conio.h>
#include<stdlib.h>
main()
{
  int gd = DETECT, gm, area, temp1, temp2, left = 25, top = 75;
  void *p;
  initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
  setcolor(YELLOW);
  circle(50, 100, 25);
  setfillstyle(SOLID_FILL, YELLOW);
  floodfill(50, 100, YELLOW);
  setcolor(BLACK);
  setfillstyle(SOLID_FILL, BLACK);
  fillellipse(44, 85, 2, 6);
  fillellipse(56, 85, 2, 6);
  ellipse(50, 100, 205, 335, 20, 9);
  ellipse(50, 100, 205, 335, 20, 10);
  ellipse(50, 100, 205, 335, 20, 11);
  area = imagesize(left, top, left + 50, top + 50);
  p = malloc(area);
  setcolor(WHITE);
  settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 2);
  outtextxy(155, 451, "Smiling Face Animation");
  setcolor(BLUE);
  rectangle(0, 0, 639, 449);
  while(!kbhit())
  {
    temp1 = 1 + random (588);
    temp2 = 1 + random (380);
    getimage(left, top, left + 50, top + 50, p);
    putimage(left, top, p, XOR_PUT);
    putimage(temp1 , temp2, p, XOR_PUT);
    delay(100);
    left = temp1;
    top = temp2;
  }
  getch();
  closegraph();
  return 0;
}
```

☞ **Output**



Lab Practicals Ends

❑❑❑