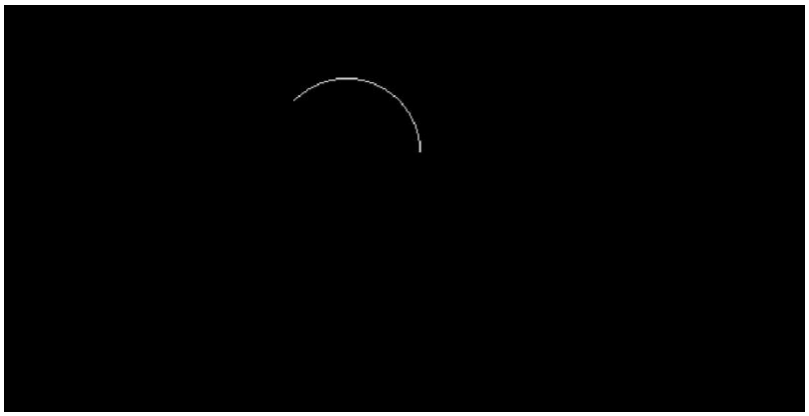
Practical No 1 (a)

Aim :Study and enlist the basic functions used for graphics in C++ and give example for each .

Code_1 : arc() function

```
#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();}
```

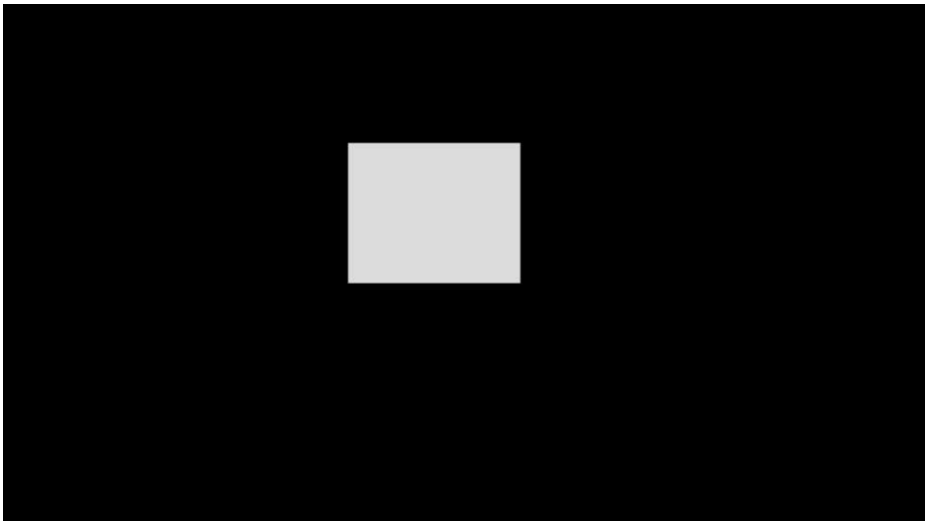
Output_1 :



Code_2 : bar

```
 #include <graphics.h>
#include <conio.h>
void main () {

int gd = DETECT , gm ;
initgraph ( &gd , &gm , "C:\\TC\\BGI"
) ;
bar ( 100 , 100 , 200 , 200 ) ; getch ()
; closegraph() ; }
```

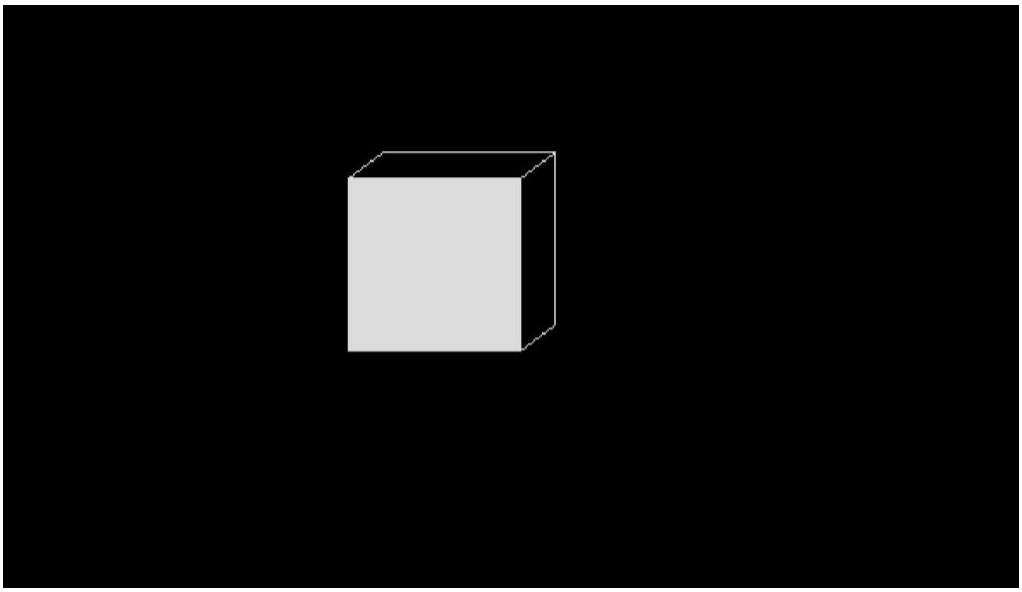
Output_2 :



Code_3 : bar3d

```
 function #include
<graphics.h> #include
<conio.h>
void main () {
int gd = DETECT , gm ;
initgraph ( &gd , &gm , "C:\\TC\\BGI"
) ;
bar3d ( 100 , 100 , 200 , 200 , 20 , 1
) ;
getch () ;
 closegraph() ;
 }
```

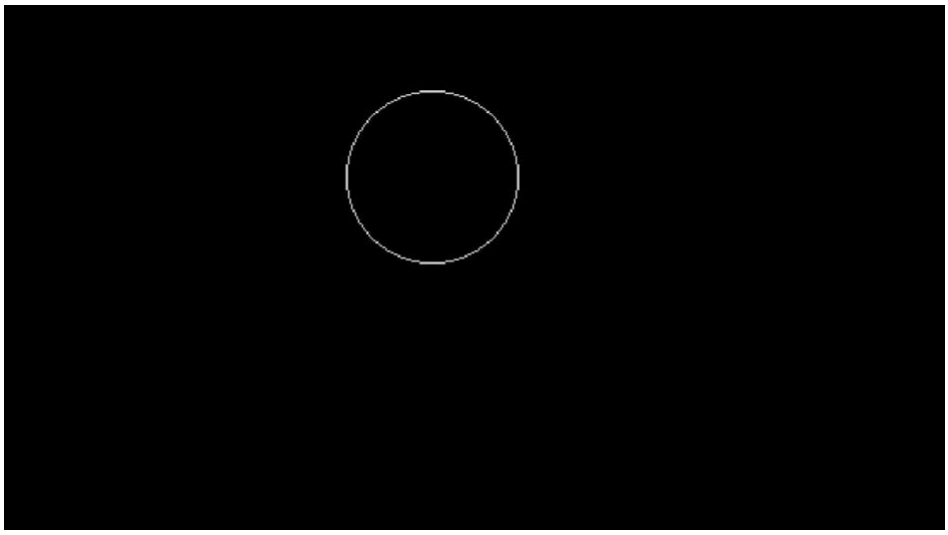
Output_3 :



Code_4 : circle

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

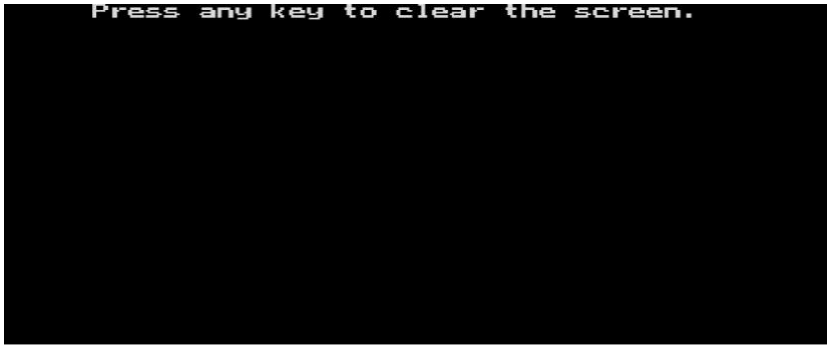
Output_4 :

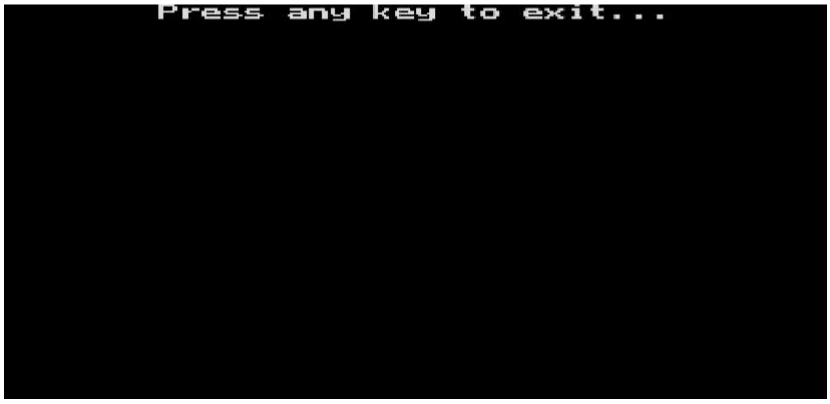


Code_5 : cleardevice

```
 function #include
<graphics.h> #include
<conio.h>
void main () {

int gd = DETECT , gm ;
 initgraph ( &gd , &gm , "C:\\TC\\BGI" ) ;
outtext ( "Press any key to clear the
screen." ) ;
getch () ;
 cleardevice();
outtext ( "Press any key to exit..." ) ;
getch();
 closegraph() ;
}
```

Output_5 :





Code_6 : closegraph

```
 function #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_6 :

Code_7 : drawpoly

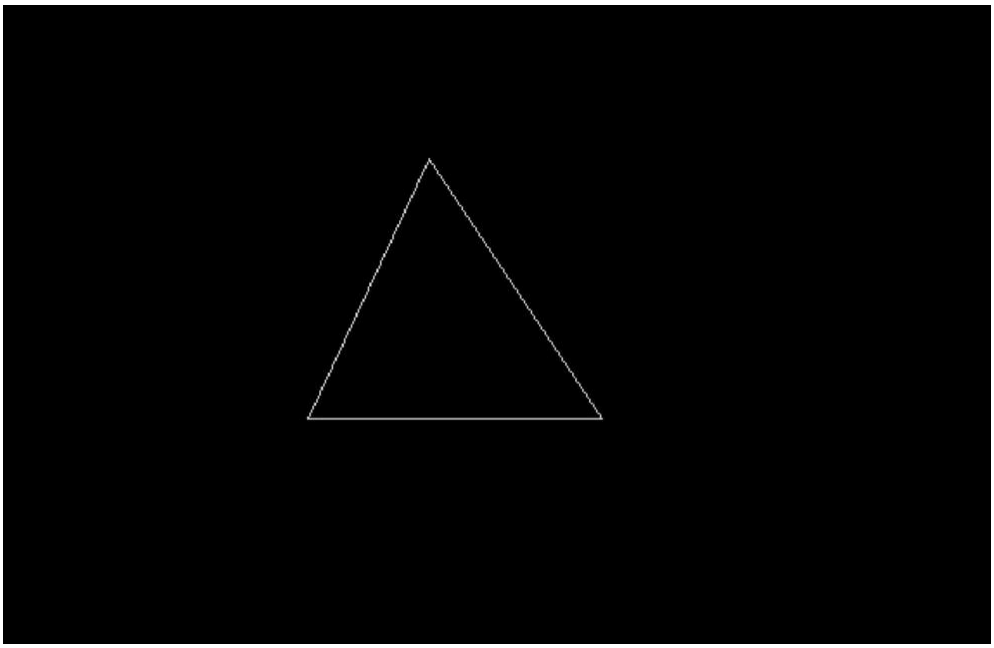
function #include

<graphics.h> #include

<conio.h>

```
 void main () {
int gd = DETECT , gm ;
initgraph ( &gd , &gm , "C:\\TC\\BGI"
) ;
ellipse ( 100 , 100 , 0 , 360 , 50 , 25 )
; getch () ; closegraph() ; }
```

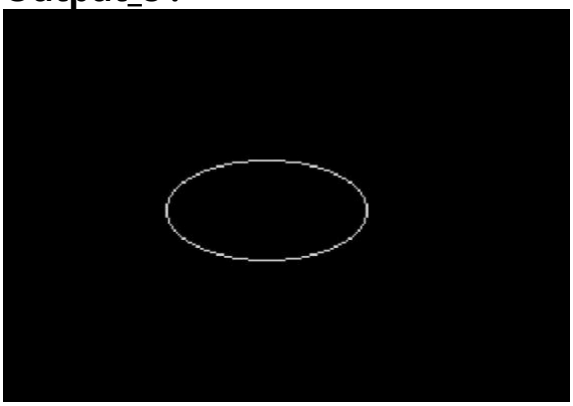
Output_7 :



Code_8 : ellipse

```
 function #include
<graphics.h> #include
<conio.h>
void main () {
int gd = DETECT , gm ;
initgraph ( &gd , &gm , "C:\\TC\\BGI" ) ;
ellipse ( 100 , 100 , 0 , 360 , 50 , 25 ) ;
getch () ;
 closegraph() ;
}
```

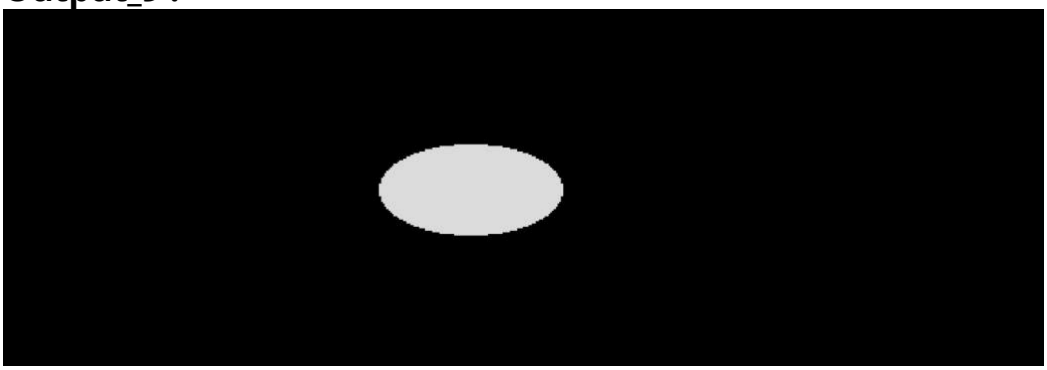
Output_8 :



Code_9 : fillellipse

```
function #include
<graphics.h> #include
<conio.h>
void main () {
int gd = DETECT , gm ;
initgraph ( &gd , &gm , "C:\\TC\\BGI" ) ;
fillellipse ( 100 , 100 , 50 , 25 ) ;
getch () ;
closegraph() ;
 }
```

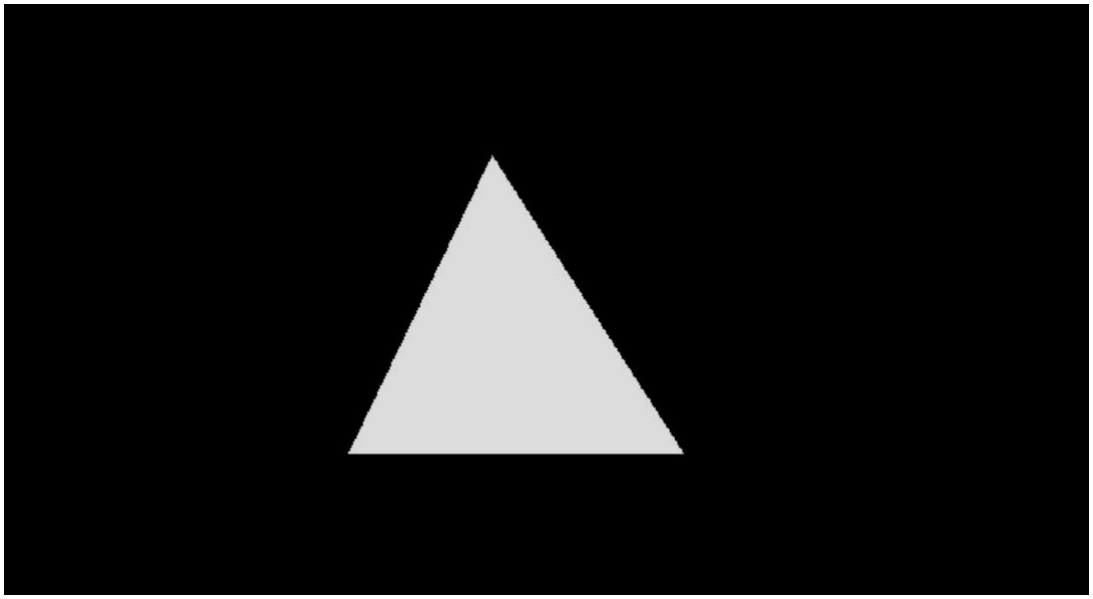
Output_9 :



Code_10 : fillpoly

```
 function #include
<graphics.h> #include
<conio.h>
void main ()
 {
int gd = DETECT , gm , points[] = { 320 , 150 , 440 , 340 , 230 , 340 , 320 , 150};
initgraph ( &gd , &gm , "C:\\TC\\BGI" ) ;
fillpoly ( 4 , points ) ;
getch () ;
closegraph() ;
}
```

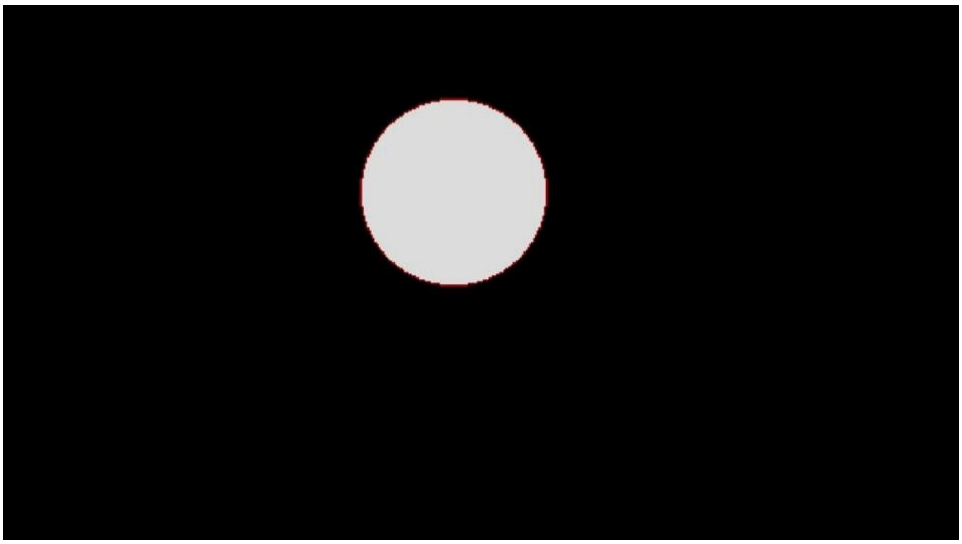
Output_10 :



Code_11 : floodfill

```
function #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_11 :

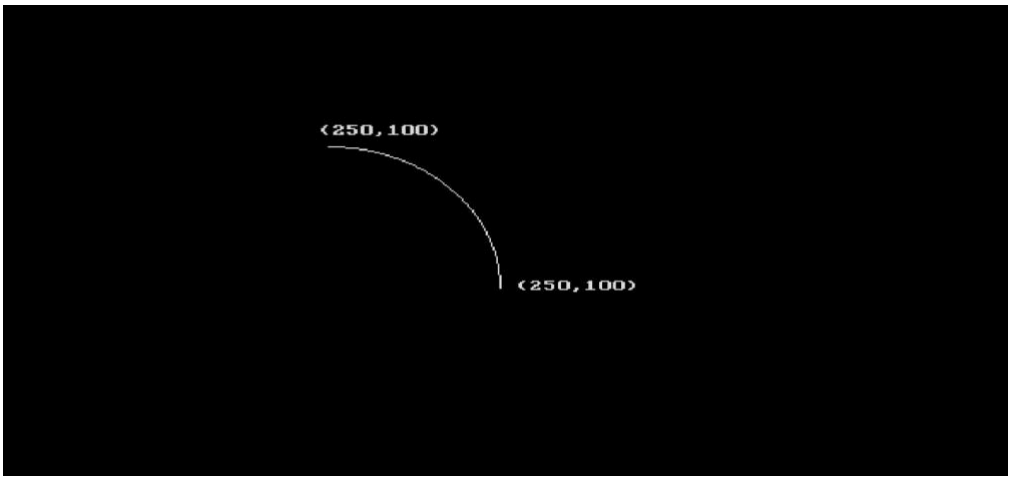


Code_12 :

getarccoords() function

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
void main () {
int gd = DETECT , gm ;
struct arccoordstype a ; char arr [100] ;
 initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
arc ( 250 , 200 , 0 , 90 , 100 ) ;
getarccoords(&a) ;
 sprintf(arr , "(%d,%d)",a.xend,a.yend);
 outtextxy(360,195,arr);
sprintf(arr , "(%d,%d)",a.xend,a.yend);
outtextxy(245,85,arr);
getch () ;
closegraph() ;
 }
```

Output_12 :



Code_13 : getbkcolor()

```
function #include
<graphics.h> #include
<conio.h> #include
<stdio.h>
void main () {
int gd = DETECT , gm , bkcolor ;

char a [100] ; initgraph ( &gd , &gm , "C:\\TC\\BGI") ;

bkcolor = getbkcolor ( ) ;

sprintf(a , "Current background color = %d" , bkcolor);

outtextxy(10,10,a);

 getch () ;

closegraph() ;

 }
```

Output_13 :

Code_14 : getcolor()

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
void main () {
int gd = DETECT , gm , drawing_color;
char a [100] ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
drawing_color = getcolor ( ) ;
sprintf(a , "Current drawing color = %d"drawing_color);
outtextxy(10,10,a);
 getch () ;
closegraph() ;
 }
```

Output_14 :

Code_15 : getdrivername()

```
 #include <graphics.h>
#include <conio.h>
#include <stdio.h>
 void main () {
int gd = DETECT , gm , drawing_color;
char *drivername;
 initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
 drivername = getdrivername ( ) ;
outtextxy(200,200,drivername);
 getch () ;
closegraph() ;
 }
```

Output_15 :

Code_16:getimage()

```
#include <graphics.h>
#include<conio.h>
#include<stdlib.h>
#include <dos.h>
void main() {
int gd = DETECT , gm , area , temp1, temp2 , left = 25 , top = 75 ;
void*p;
 initgraph ( &gd , &gm , "C:\\TC\\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(0,0,639,4
49); while(!kbhit()){
```

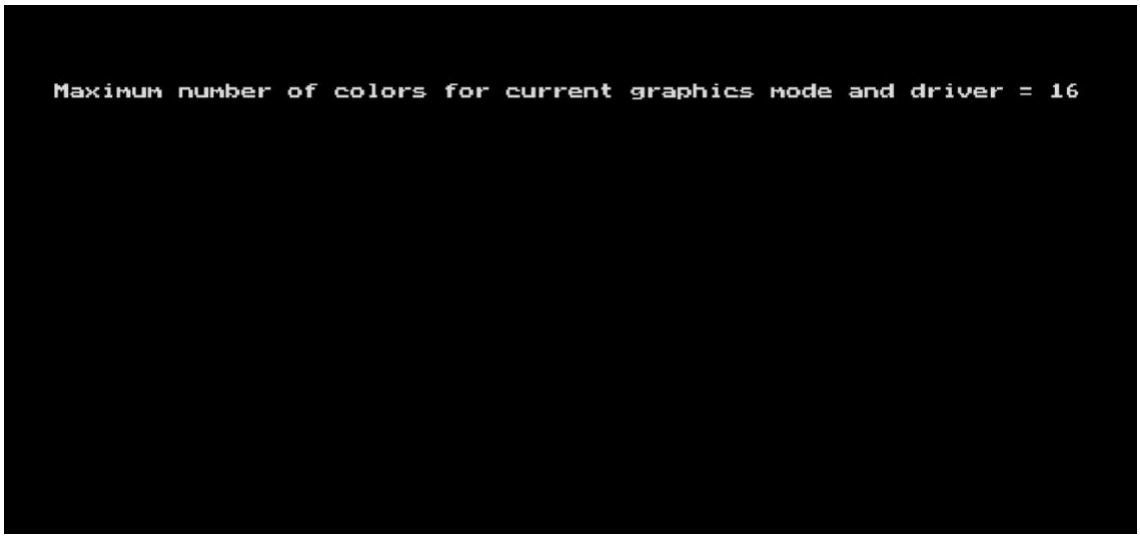
```
temp1 = 1 + random (588); temp2 =
1 + random (380); getimage(left,
top, left + 50, top + 50, p); putimage
(left, top, p,XOR_PUT);
putimage(temp1, temp2,
p,XOR_PUT);
delay(100);
left = temp1;
top=temp2;}
 getch () ;
 closegraph() ;
 }
```

Output_16 :

Code_17 : getmaxcolor() function

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
void main () {
int gd = DETECT , gm , max_colors;
char a [100] ; initgraph ( &gd , &gm , "C:\\TC\\BGI") ; max_colors =
getmaxcolor ( ) ; sprintf(a , "Maximum number of colors for current
graphics mode and driver = %d" ,
max_colors+1);
outtextxy(0,40,
a);
getch () ;
closegraph() ; }
```

Output_17 :



Code_18 : getmaxx() function

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
 void main () {
int gd = DETECT , gm , max_x;
char array [100] ; initgraph ( &gd , &gm , "C:\\TC\\BGI") ; max_x = getmaxx
( ) ; sprintf(array , "Maximum X coordinate for current graphics mode and
driver = %d" ,

max_x)
outtext(array);
getch () ;
closegraph() ; }
```

Output_18 :

Code_19 : getmaxy() function

```
#include <graphics.h>
#include <conio.h>
 #include <stdio.h>
void main () {
int gd = DETECT , gm , max_y;
char array [100] ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
max_y = getmaxy ( ) ;
sprintf(array , "Maximum Y coordinate for current graphics mode and driver = %d" ,
max_y);
outtext(array);
getch () ;
closegraph() ; }
```

Output_19 :

Code_20 : getpixel() function

```
 #include <graphics.h>
#include <conio.h>
#include <stdio.h>
 void main () {

int gd = DETECT , gm , color;

char array [50] ;

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

color = getpixel (0,0) ;

sprintf(array , "color of pixel at (0,0) = %d" , color);

outtext(array);

getch () ;

closegraph() ; }
```

Output_20 :

Code_21 : getx() function

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
void main () {

int gd = DETECT , gm , x ;

char array [100] ;

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

sprintf(array , "Current position of x = %d" , getx() );

outtext(array);

getch () ;

closegraph() ; }
```

Output_21 :

Code_22 : gety() function

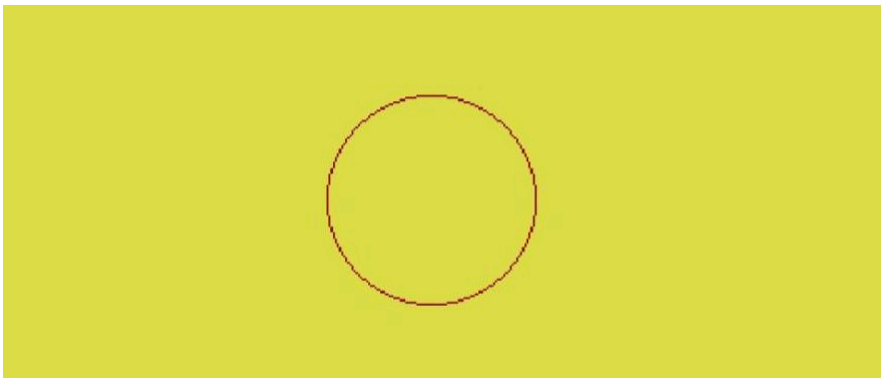
```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
void main () {
int gd = DETECT , gm , y;
char array [100] ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
y = gety ( ) ;
sprintf(array , "Current position of y = %d" , y);
outtext(array);
getch () ;
closegraph() ; }
```

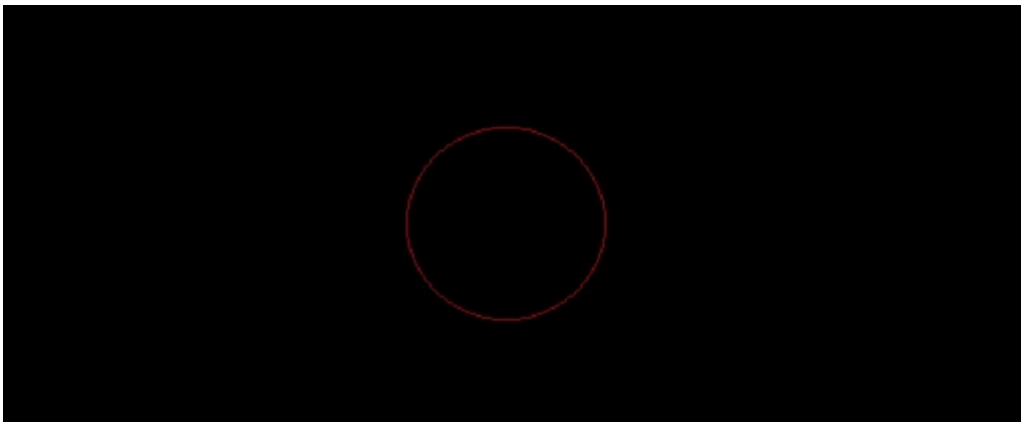
Output_22 :

Code_23 : graphdefaults() function

```
 #include <graphics.h>
#include <conio.h>
 void main () {
int gd = DETECT , gm ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
setcolor (RED) ;
setbkcolor (YELLOW) ;
circle (250 , 250 , 50) ;
getch () ;
graphdefaults() ;
getch () ;
closegraph() ; }
```

Output_23 :





Code_24 : grapherrormsg() function

```
#include<graphics.h>
#include<conio.h>
#include<stdlib.h>
#include <stdio.h>
void main () {
int gd = DETECT , gm , errorcode;
char array [100] ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
errorcode = graphresult ( ) ;
if ( errorcode != grOk ) {
printf ( "Graphics error : %s\n" , grapherrormsg (errorcode)) ;
printf ( "Press any key to exit." ) ;
getch();
exit(1) ; }
getch () ;
closegraph() ; }
```

Output_24 :

Code_25:imagesize() function

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main () {
int gd = DETECT , gm , bytes;
char array [100] ; initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
circle (200 , 200 , 50) ; line (150 , 200 , 250 , 200) ;
line (200 , 150 , 200 , 250) ;
sprintf(array , "Number of bytes required to store required area = %d" ,
bytes);
 outtextxy (10 , 280 , array);
 getch () ;
closegraph() ;
 }
```

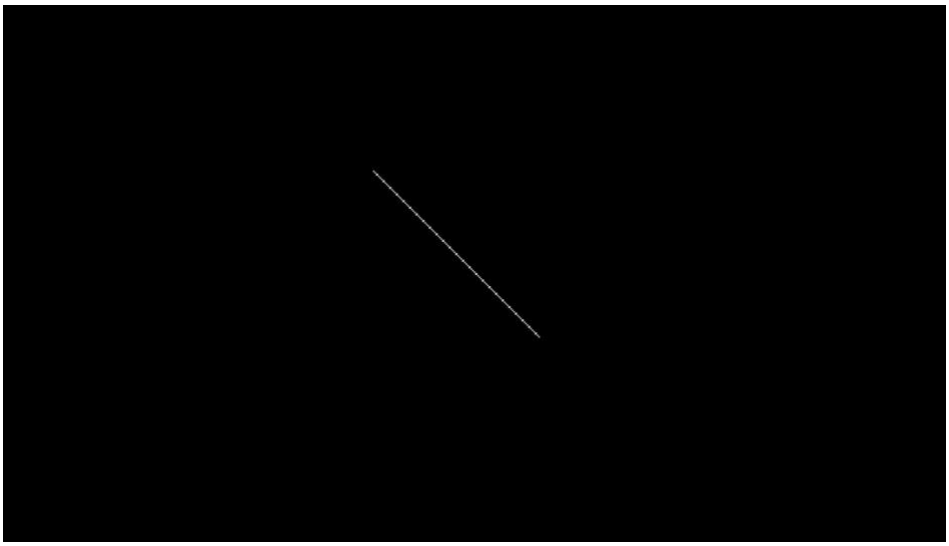
Output_25 :

Code_26 :

line() function

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

Output_26 :

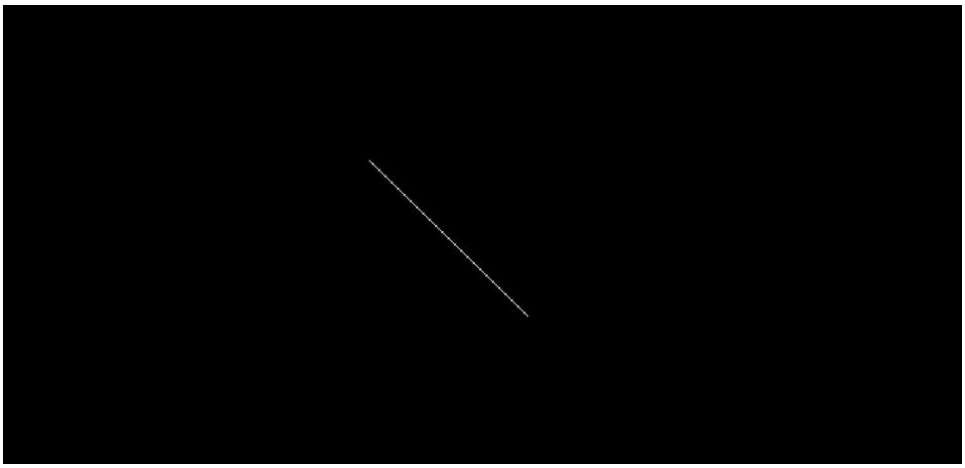


C**ode_27 :**

lineto() function

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

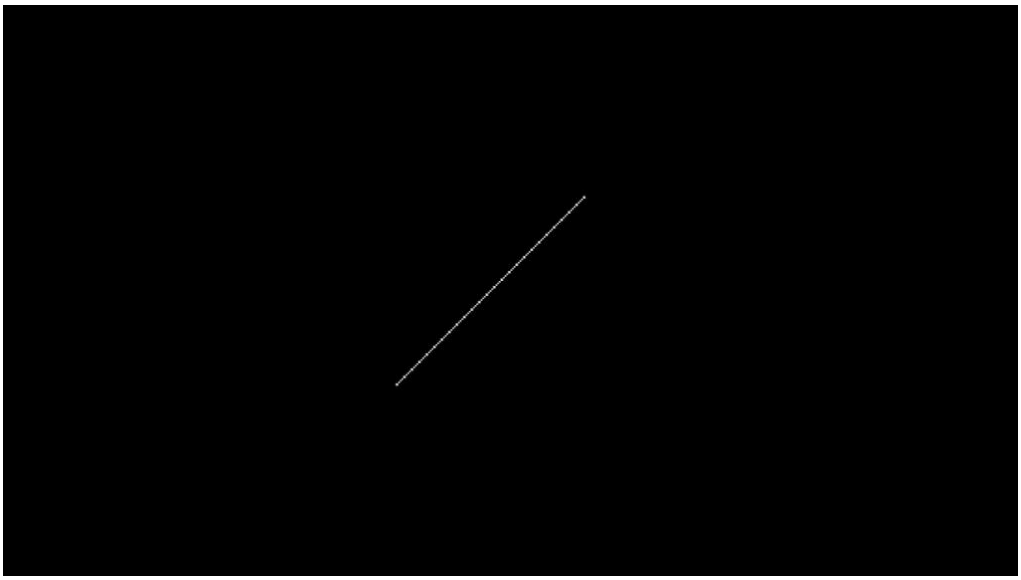
Output_27 :



Code_28 :linerel() function

```
#include <graphics.h>
#include <conio.h>
void main () {
int gd = DETECT , gm ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
moveto (250 , 250 ) ;
linerel (100 , -100 ) ;
getch () ;
closegraph() ; }
```

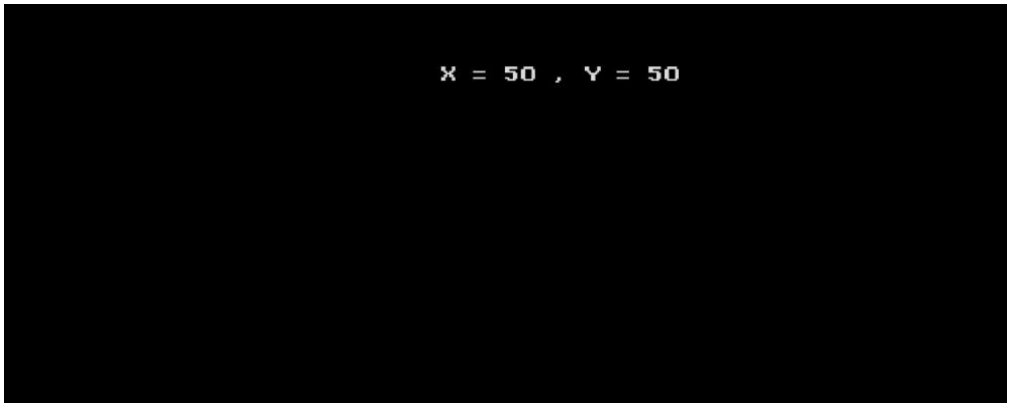
Output_28 :



Code_29 : moveto() function

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
 void main () {
int gd = DETECT , gm , x , y ;
char msg [100] ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
sprintf(msg , "X = %d , Y = %d" , getx() , gety() );
outtext(msg);
moveto (50,50) ;
sprintf(msg , "X = %d , Y = %d" , getx() , gety() );
outtext(msg);
getch () ;
closegraph() ; }
```

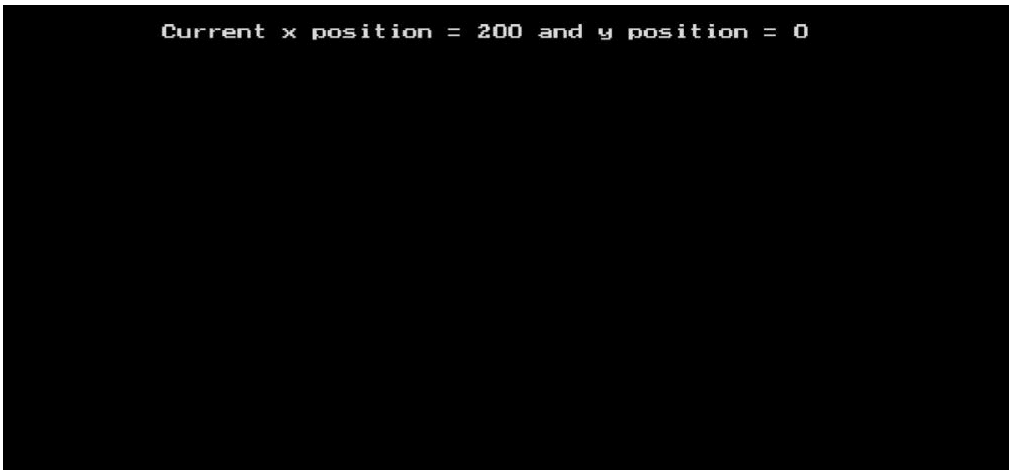
Output_29 :



Code_30 : moverel() function

```
 #include <graphics.h>
#include <conio.h>
#include <stdio.h>
void main () {
int gd = DETECT , gm , x , y ;
char message [100] ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
moveto (100,100) ;
moverel (100,-100) ;
x = getx ( ) ; y = gety ( ) ;
 sprintf(message , "Current x position = %d and y position = %d" , x ,
y);
outtextxy(10 , 10 , message);
getch () ;
 closegraph() ; }
```

Output_30 :

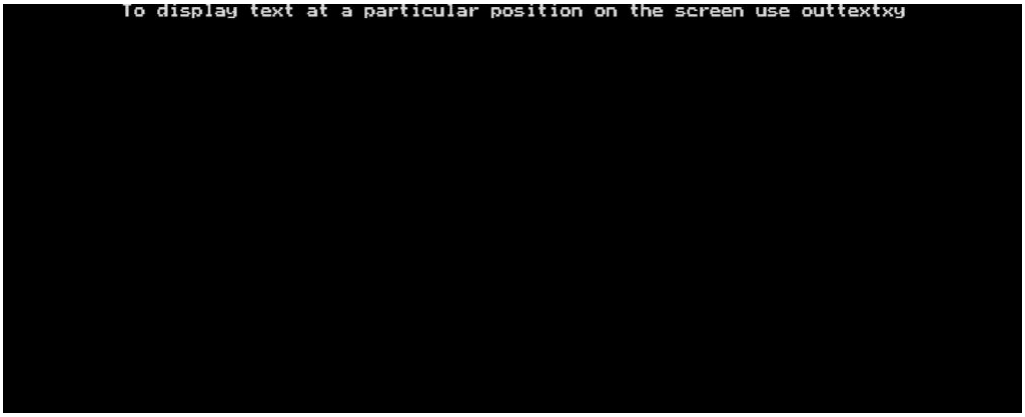


Code_31 :

outtext() function

```
 #include <graphics.h>
#include <conio.h>
void main () {
int gd = DETECT , gm ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
outtext ( "To display text at a particular position on the screen use outtextxy" ) ;
getch () ;
closegraph() ; }
```

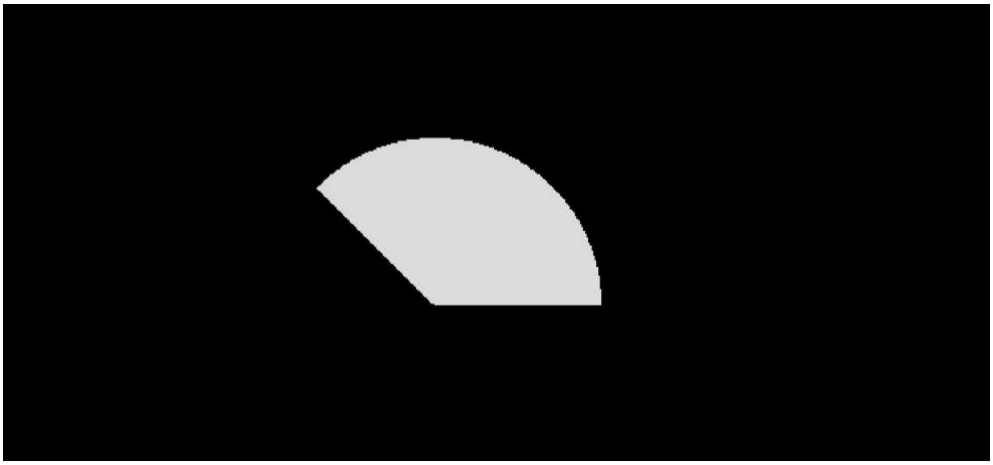
Output_31 :



Code_32 : pieslice() function

```
#include <graphics.h>
#include <conio.h>
void main () {

int gd = DETECT , gm ;

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

pieslice (200 , 200 , 0 , 135 , 100 ) ;

getch () ;

closegraph() ; }
```

Output_32 :



Code_33:putimage() function

```
#include<graphics.h>
#include<conio.h>
#include<stdlib.h>
#include<dos.h>
void main()          {

int gd = DETECT, gm, area, temp1, temp2, left = 25, top =

75;

void]*p;

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

Output_33 :

Code_34 :

putpixel() function

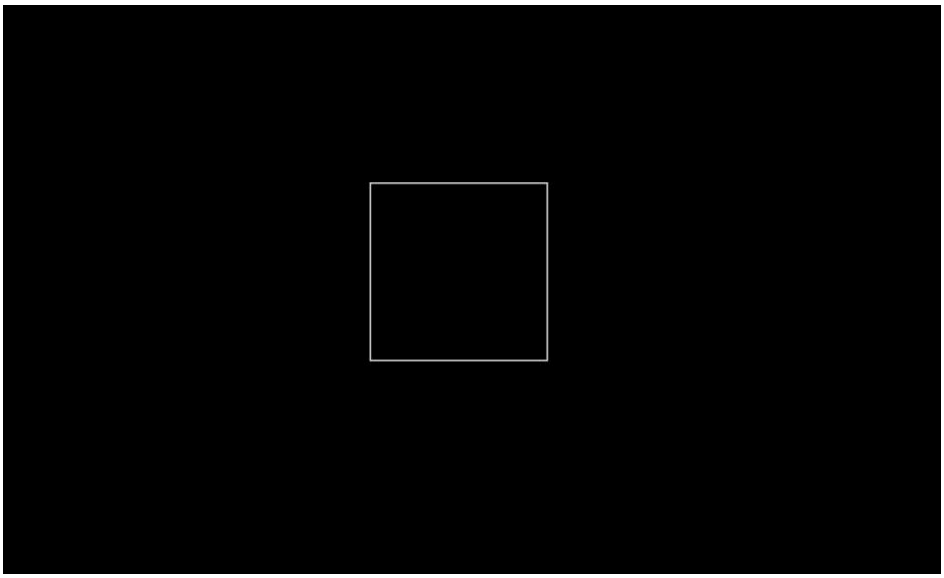
```
#include <graphics.h>
#include <conio.h>
void main () {
int gd = DETECT , gm ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
putpixel (25 , 25 , RED ) ;
getch () ;
closegraph() ; }
```

Output_34 :.

Code_35:rectangle() function

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

Output_35 :

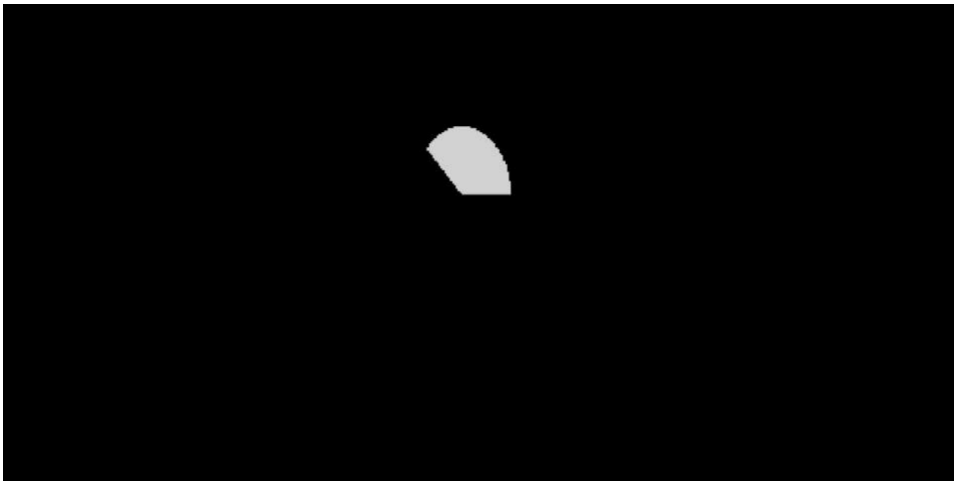


Code_36 :

sector () function

```
#include <graphics.h>
#include <conio.h>
 void main () {
int gd = DETECT , gm ;
initgraph ( &gd , &gm , "C:\\TC\\BGI") ;
sector (100 , 100 , 0 , 135 , 25 , 35 ) ;
getch () ;
closegraph() ; }
```

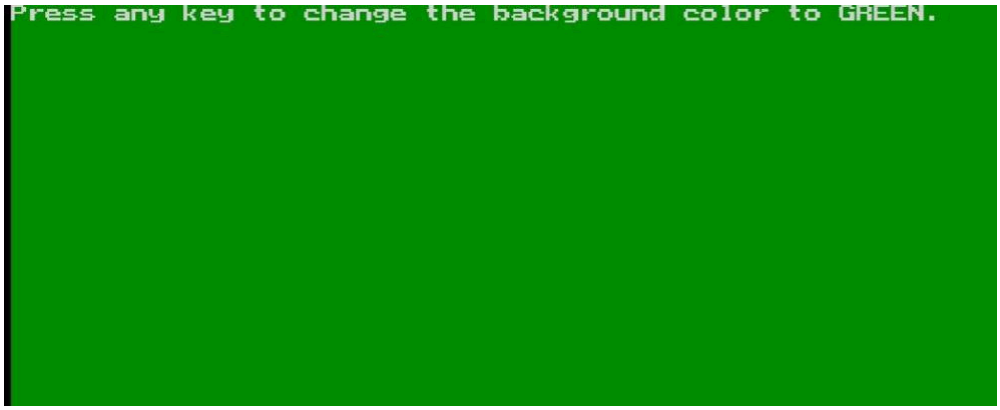
Output_36 :



Code_37 : setbkcolor() function

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

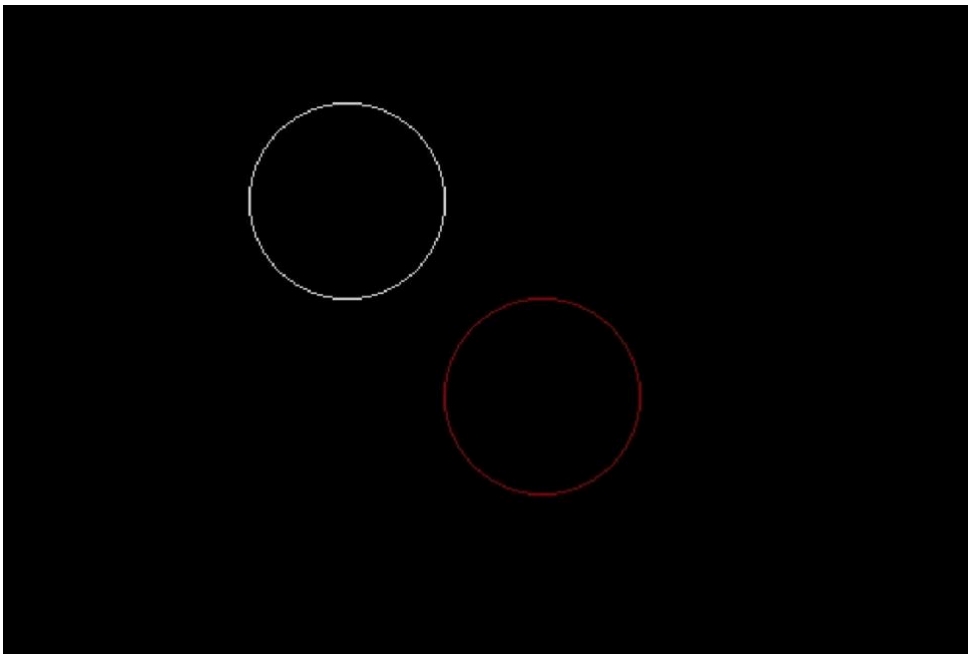
Output_37 :



Code_38 : setcolor() function

```
#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() ; }
```

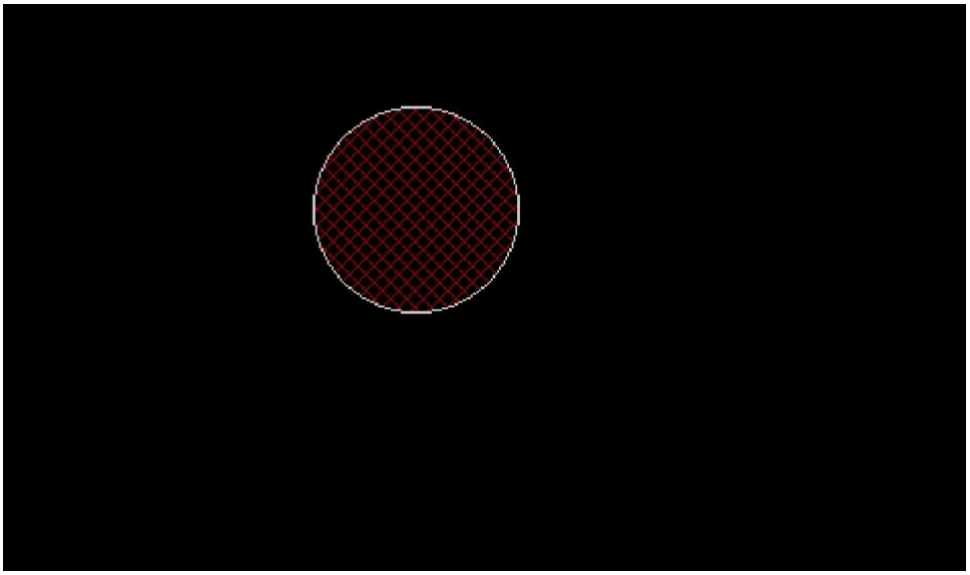
Output_38 :



Code_39 : setfillstyle() function

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

Output_39 :



Code_40 : setlinestyle()
function

```
 #include <graphics.h>
#include <conio.h>
 void main(){

int gd = DETECT , gm, c, x = 100, y = 50;

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

for (c = 0; c <5; c++){

setlinestyle(c, 0, 2);

line(x, y, x+200, y);

y = y + 25;}

 getch();

 closegraph();}
```

Output_40 :

Code_41 : settextstyle() function

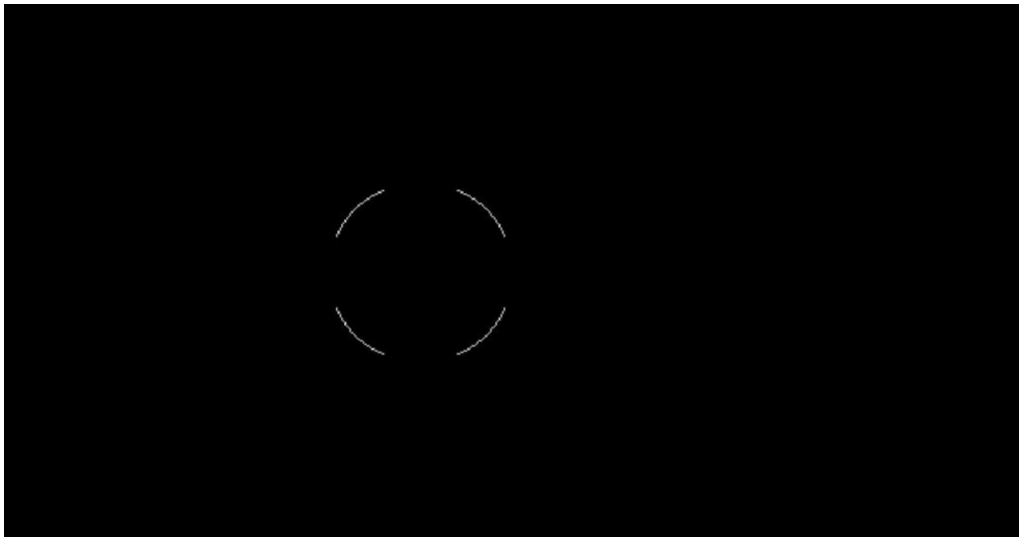
```
#include <graphics.h>

#include<conio.h>

void main(){

int gd = DETECT, gm, x = 25, y = 25, font = 0;

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

for (font = 0; font <= 10; font++) {
 settextstyle(font, HORIZ_DIR, 1);

 outtextxy(x, y, "Text with different fonts");

 y = y + 25;}

  getch();

  closegraph();}
```

Output_41 :

Code_42 : setviewport() function

```
#include<graphics.h>
#include<conio.h>
void main(){
int gd = DETECT, gm, midx, midy;
initgraph(&gd, &gm, "C:\\TC\\BGI");
midx=getmaxx()/2;
midy = getmaxy()/2;
setviewport(midx - 50, midy - 50, midx + 50,midy +50, 1);
circle(50, 50, 55);
getch();
closegraph();}
```

Output_42 :



Code_43 : textheight() function

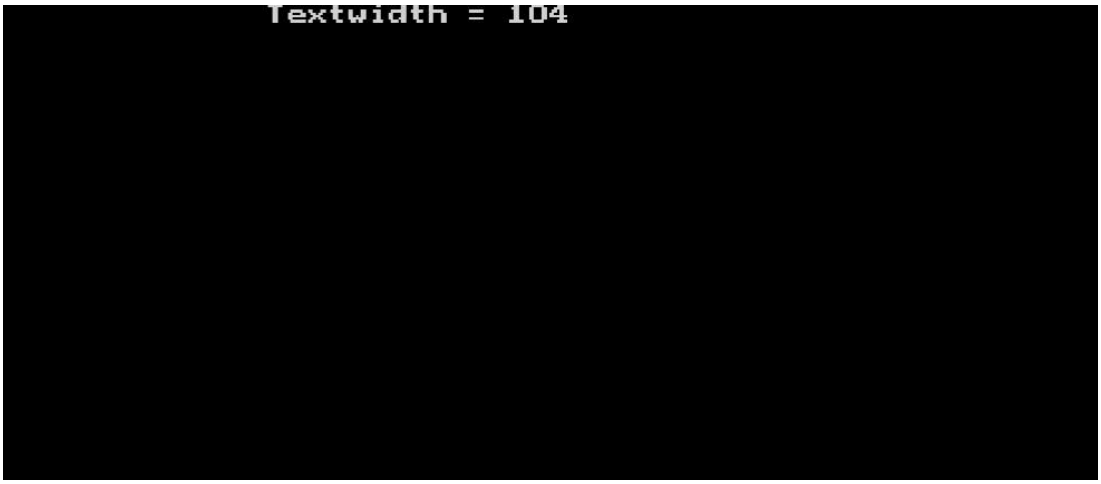
```
#include <graphics.h>
#include<conio.h>
#include<stdio.h>
 void main(){
int gd = DETECT, gm, height;
char array[100];
initgraph(&gd, &gm, "C:\\TC\\BGI");
height = textheight("C programming");
sprintf(array , "Textheight = %d",height);
outtext(array);
getch();
closegraph();}
```

Output_43 :

Code_44 : textwidth() function

```
 #include <stdio.h>
#include <graphics.h>
#include <conio.h>
void main(){

int gd = DETECT, gm, width;

char array[100];

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

width  =  textwidth("C  programming");

sprintf(array,"Textwidth= %d",width);

outtext(array);

 getch();

 closegraph();}
```

Output_44 :



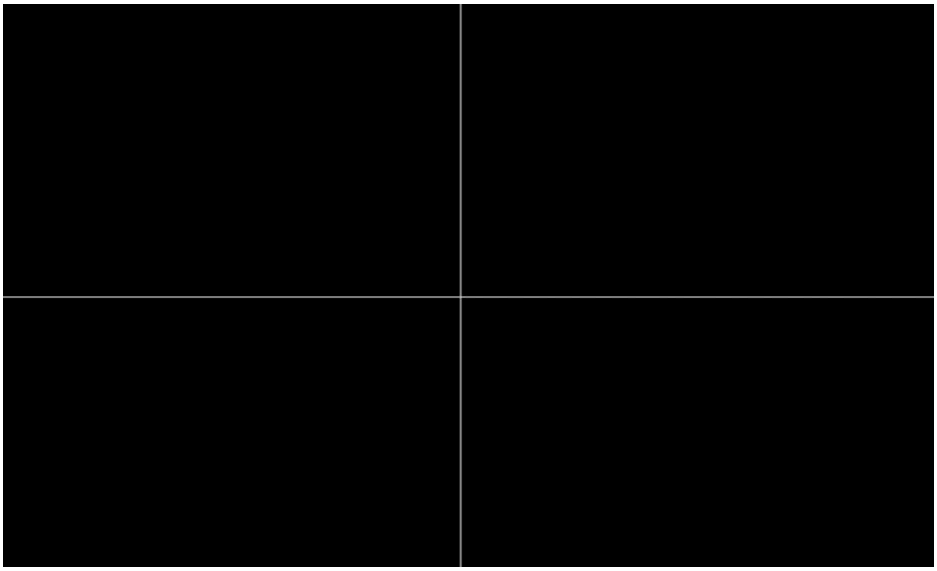
Practical No 1 (b)

Aim : Draw co-ordinate axis at the centre of the screen.

Code :

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
int midx,midy;
initgraph(&gd,&gm,"C:\\TC\\BGI");
cleardevice();
midx=getmaxx()/2;
midy=getmaxy()/2;
line(1,midy,640,midy);
line(midx,1,midx,480);
getch(); }
```

Output :

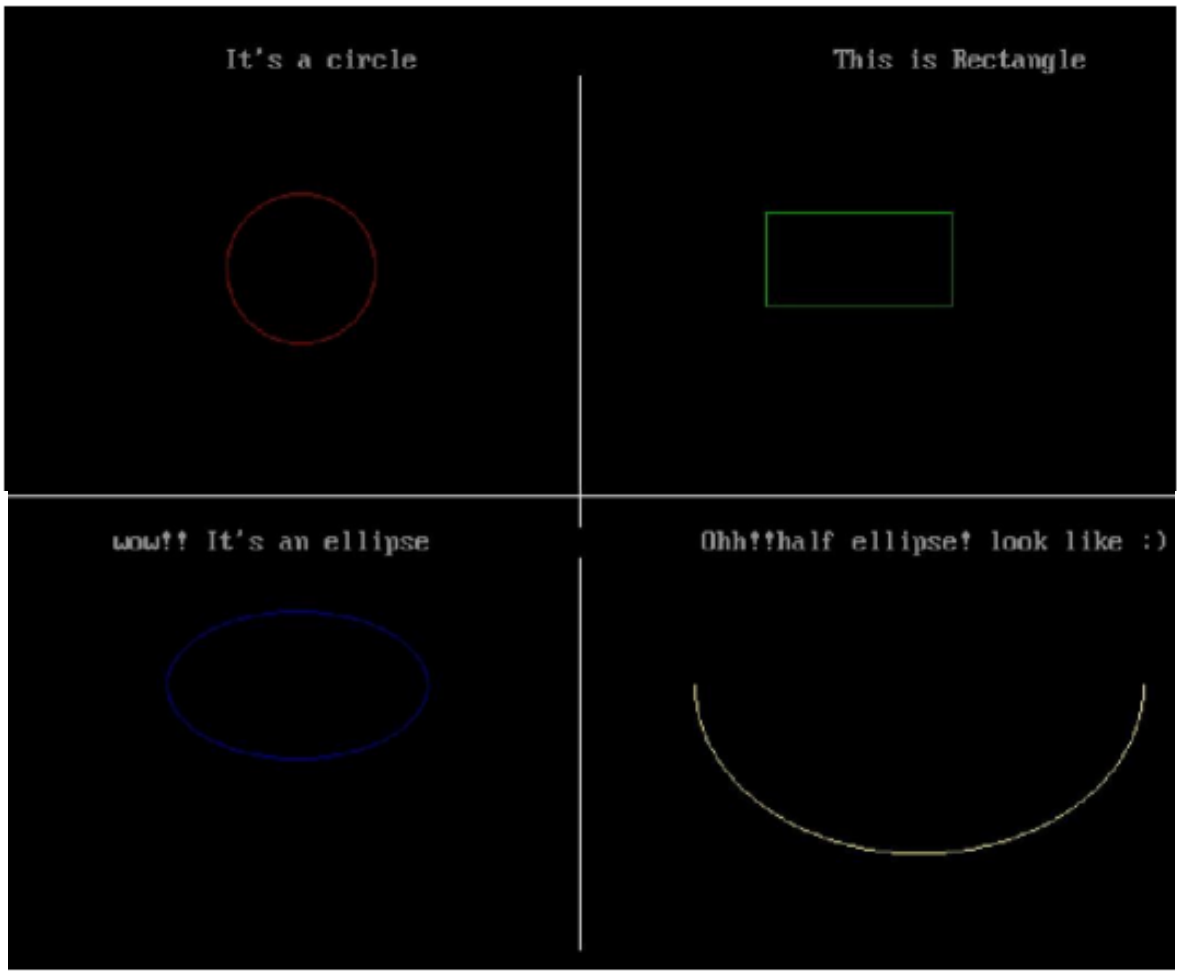


Practical No 2(a)

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

```
#include <stdio.h>
#include<conio.h>
#include<graphics.h>
void main(){
        int gd = DETECT , gm;
        int midx , midy ;
        initgraph ( &gd , &gm , "C:\\TC\\BGI" ) ;
        cleardevice() ;
        midx=getmaxx()/2;
        midy=getmaxy()/2;
        //coordinate Axis
        line(1,midy,640,midy);
        line(midx,1,midx,480);
        setcolor(RED);
        circle(midx+(-150),midy-(120),40);
        printf("\t\tIt's a circle ");
        setcolor(GREEN);
        rectangle(midx+(100),midy-(100),midx+(200),midy-(150));
        printf("\t\t\t\t This is Rectangle\n\n\n\n\n");
        setcolor(BLUE);
        ellipse(midx+(-150),midy-(-100),0,360,midx+(- 250),midy-(200));
        printf("\n\n\n\n\n\n\n\n\n\n\n\t wow!! It's an ellipse");
        setcolor(YELLOW);
         ellipse(midx+(180),midy-(-100),180,0,midx+(-200),midy- (150));
         printf("\t\t\t Ohh!!half ellipse! looks like :)");
         getch();}
```

Output :

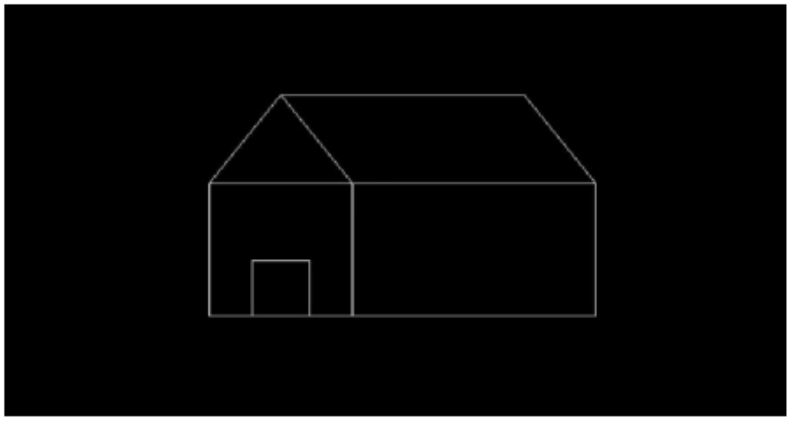

It's a circle
This is Rectangle
wow!! It's an ellipse
Ohh!!half ellipse! look like :)

Practical No 2 (b)

Aim : Draw simple and colorful hut**.**

```
#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(200,100,370,100);
        line(370,100,420,180);
        getch();
        closegraph();}
```

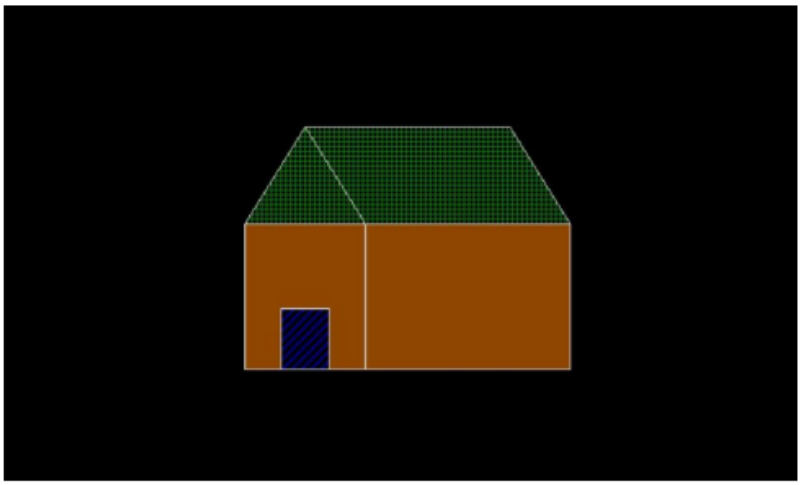
Output:



Code_2 : Colourful hut

```
#include<graphics.h>
#include<conio.h>
void main(){
        int gd = DETECT , gm ;
        initgraph(&gd, &gm, "C:\\TC\\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);
        /*Fill colours*/
        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

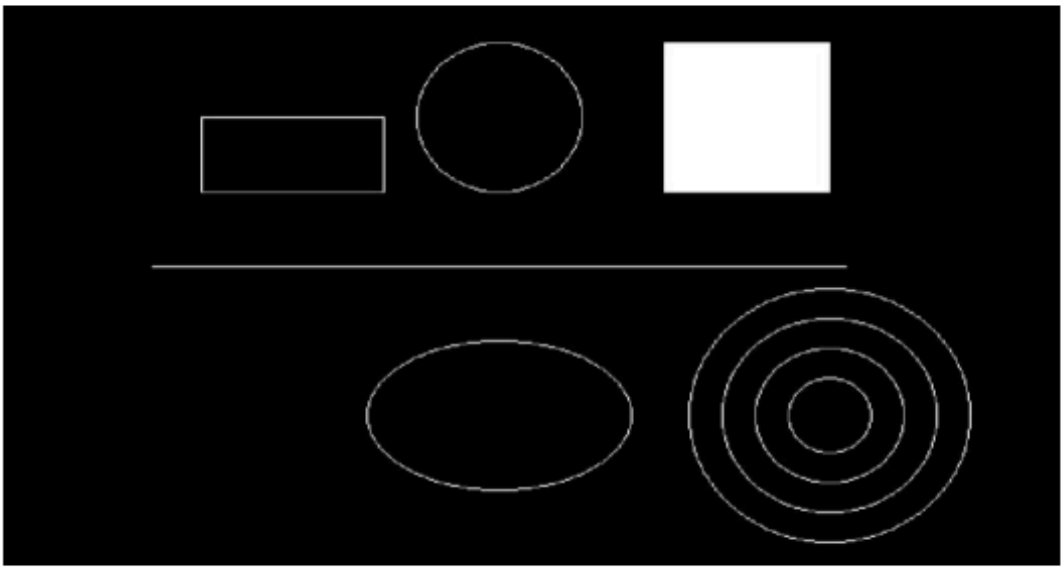


Practical No. 3

Aim : Draw the basic shapes on screen.

```
#include <graphics.h>

#include<conio.h>

void main() {

        int gd= DETECT , gm , left=100 , top=100 , right=200 , bottom=200 , x =300 , y=150 ,
radius=50 ;

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

        rectangle(120, 150, 230, 200);

        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, 80, 50);

        for (radius = 25; radius <= 100; radius = radius +20)

        circle(500,350,radius) ;

        getch() ;

        closegraph() ; }
```

Output :



Practical No.

4(A) **Aim :** Develop a program for DDA Line drawing algorithm. **Code :**

```
<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,pixel;
        int i,gd,gm;
        printf("Enter the value of x1 :
        "); scanf("%f",&x1);
        printf("Enter the value of y1 :
        "); scanf("%f",&y1);
        printf("Enter the value of x2 :
        "); scanf("%f",&x2);
        printf("Enter the value of y2 : ");
        scanf("%f",&y2);
        detectgraph(&gd,&gm);
        initgraph(&gd,&gm,"C:\\TC\\BGI"
        ); dx=abs(x2-x1);
        dy=abs(y2-y1);
        if(dx>=dy)
        pixel=dx;
        else
        pixel=dy;
```

```
dx=dx/pix
el;

dy=dy/pixel;
x=x1 ;
y=y
1 ;
i=1;
while
  (i<=pixel) {
  putpixel(x,y,1)
  ; x=x+dx;
  y=y+dy;
  i=i+1;
  delay(100
  );}
getch() ;
closegraph () ;
}
```

Output :

Practical No. 4(B)

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

Code :

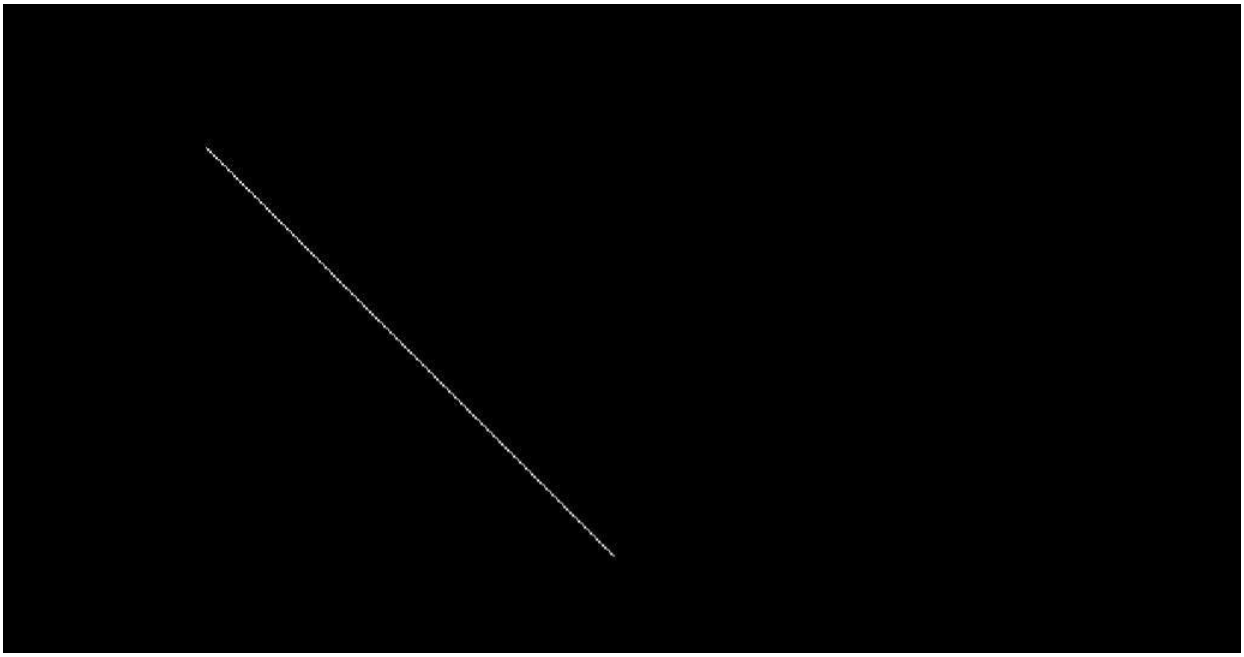
```
#include <graphics.h>
#include <stdio.h>
#include
<conio.h>
#include
```

```
<math.h>     void
main() {
        int
        x,y,x1,y1,x2,y2,dx,dy,p;
        int gd,gm;
        clrscr() ;
        printf("\n\nEnter the coordinates of first point : ");
        scanf("%d%d",&x1,&y1);
        printf("\n\nEnter the coordinates of second point : ");
        scanf("%d%d",&x2,&y2);
        dx = (x2-x1);
        dy=(y2-y1);
        p=2 *(dy)
        *(dx); x=x1;
        y=y1;
        detectgraph(&gd,&gm);
        initgraph(&gd,&gm,"C:\\TC\\BGI"
        ); putpixel(x,y,WHITE);
        while(x <= x2) {
                if(p < 0) {
                        x=x+
                        1;
                        y=y;
                        p=p+2*(dy); }
                                                                        else {

x=x+1; y=y+1;
p=p+2 *(dy - dx); }

                putpixel(x,y,WHITE)
```

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

```
Enter the coordinates of first point : 100
200

Enter the coordinates of second point : 300
400
```

Output :



Practical No. 5(A)

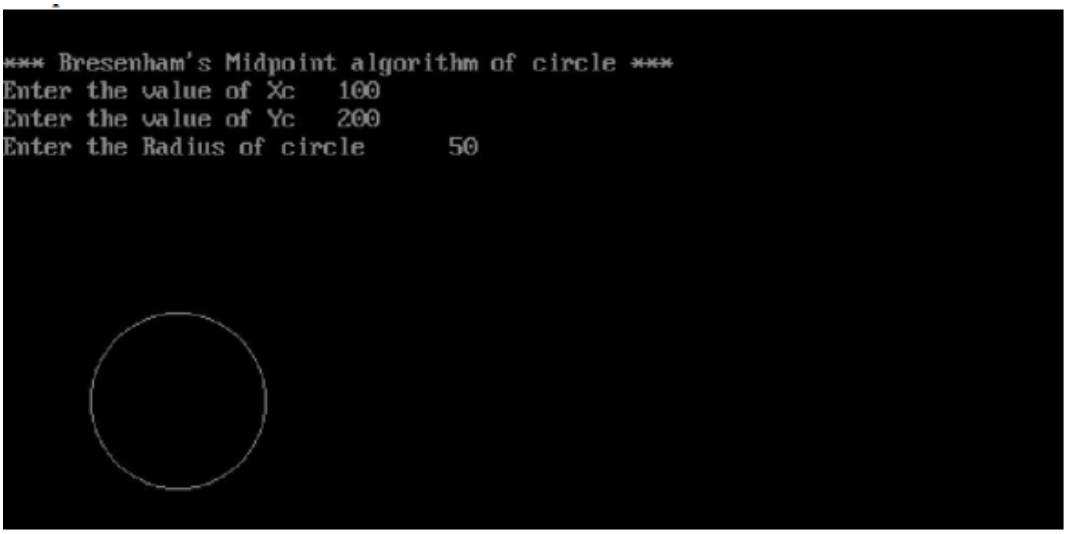
Aim : Develop a program for the mid point circle drawing algorithm.

Code :

```
#include<stdio.h>
#include<conio.h>
#include <graphics.h>
void pixel(int xc,int yc,int x,int y);
void main() {
        int gd=DETECT,gm,xc,yc,r,x,y,Pk;
        clrscr();
        initgraph(&gd,&gm, "C:\\TC\\BGI");
         printf("*** Bresenham's Midpoint algorithm of circle ***\n");
        printf("Enter the value of Xc\t");
        scanf("%d",&xc);
        printf("Enter the value of Yc \t");
        scanf("%d",&yc);
        printf("Enter the Radius of circle\t");
        scanf("%d",&r);
        x=0;
        y=r;
        Pk=1-r;
        pixel(xc,yc,x,y);
        while(x<y) {
                if(Pk<0) {
                        x=x+1;
                        Pk=Pk+(2*x)+1; }
                else {
```

```
                    x=x+1;
                    y=y-1;
                    Pk=Pk+(2*x)-(2*y)+1; }
            pixel(xc,yc,x,y); }
    getch () ;
    closegraph(); }
void pixel(int xc,int yc,int x, int y) {
    putpixel(xc+x,yc+y,7);
    putpixel(xc+y,yc+x,7);
    putpixel(xc-y,yc+x,7);
    putpixel(xc-x,yc+y,7);
    putpixel(xc-x,yc-y,7);
    putpixel(xc-y,yc-x,7);
    putpixel(xc+y,yc-x,7);
    putpixel(xc+x,yc-y,7); }
```

Output:



Practical No. 5(B)

Aim : Develop a program for the mid point ellipse drawing algorithm.

Code :

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void disp();
float x,y;
int xc,yc;
void main() {
        int gd=DETECT,gm;
        int a,b;
        float p1,p2;
        clrscr();
        initgraph(&gd,&gm,"C://TC//BGI");
        printf("Enter xc:\t");
        scanf("%d",&xc);
        printf("Enter yc:\t");
        scanf("%d",&yc);
        printf("Enter a:\t");
        scanf("%d",&a);
        printf("Enter b:\t");
        scanf("%d",&b);
        x=0;
        y=b;
        disp();
```

```
p1=(b*b)-(a*a*b)+(a*a)/4;
while((2.0*b*b*x)<=(2.0*a*a*y)) {
        x++;
        if(p1<=0) {
                p1=p1+(2.0*b*b*x)+(b*b); }
        else {
                y--;
                p1=p1+(2.0*b*b*x)+(b*b)-(2.0*a*a*y); }
        disp();
        x=-x
        disp();
        x=-x; }
x=a;
y=0;
disp();
p2=(a*a)+2.0*(b*b*a)+(b*b)/4;
while((2.0*b*b*x)>(2.0*a*a*y)) {
        y++;
        if(p2>0) {
                p2=p2+(a*a)-(2.0*a*a*y); }
        else {
                x--;
                p2=p2+(2.0*b*b*x)-(2.0*a*a*y)+(a*a); }
        disp();
        y=-y;
        disp();
        y=-y; }
getch();
```

```
        closegraph (); }
void disp() {
putpixel (xc+x,yc+y,10);
putpixel (xc-x,yc+y,10);
putpixel (xc+x,yc-y,10);
putpixel (xc+x,yc+y,10); }
```

Output:

Practical No 6(a)

Aim : Program to implement 2D scaling.

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main() {
        int i;
        int gd=DETECT,gm;
        int x2,y2,x1,y1,x,y;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        printf("Enter the 2 line end points: x1,y1,x2,y2:\n");
        scanf("%d\n%d\n%d\n%d", &x1,&yl,&x2,&y2);
        line(x1,y1,x2,y2); printf("\nEnter scaling co-ordinates;xit y\t \n");
        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(); }
```

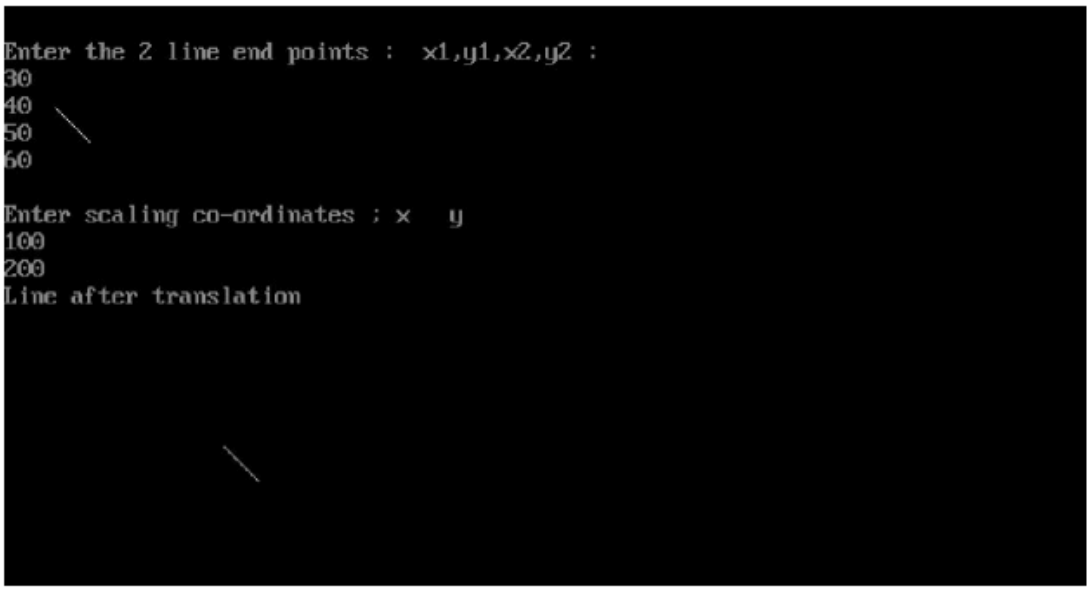
Output :

Practical No. 6(b)

Aim : Program to perform 2D translation**.**

```
#include <graphics.h>
#include<stdio.h>
#include<conio.h>
void main() {
        int i;
        int gd=DETECT,gm;
        int x2,y2,x1,y1,x,y;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        printf("Enter the 2 line end points :  x1,y1,x2,y2 : \n");
        scanf("%d\n%d\n%d\n%d",&x1,&y1,&x2,&y2);
        line(x1,y1,x2,y2);
        printf("\nEnter scaling co-ordinates ; x\t y\t \n");
        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 the 2 line end points : x1,y1,x2,y2 :
30
40
50
60
Enter scaling co-ordinates : x y
100
200
Line after translation

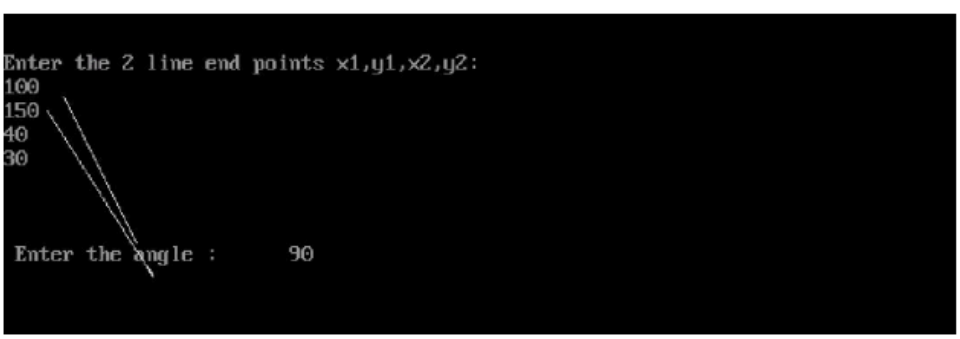
Practical No. 7(A)

Aim : Perform 2D Rotation on a given object.

Code :

```
#include<graphics.h>
#include <math.h>
#include<stdio.h>
void main(){
        int gd=DETECT,gm;
        int i;
        int x2,y2,x1,y1,x,y,xn,yn;
        double r11,r12,r21.r22,th;
        initgraph(&gd,&gm,"c:\\turboc3\\bgi");
        printf("Enter the 2 line end points x1,y1,x2,y2: \n");
        scanf("%d %d%d %d", &x1,&y1,&x2, &y2);
        line(x1,y1,x2,y2);
        printf ("\n\n\n Enter the angle : \t") ;
        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*r21));
        yn=((x2 r12)+(y2*r22));
        line(x1,y1,xn,yn);
        getch();
        closegraph() ; }
```

Output :



Practical No. 7(B)

Aim : Program to create a house like figure and perform the following operation

Code :

```
#include < graphics.h >
#include < stdio.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);
```

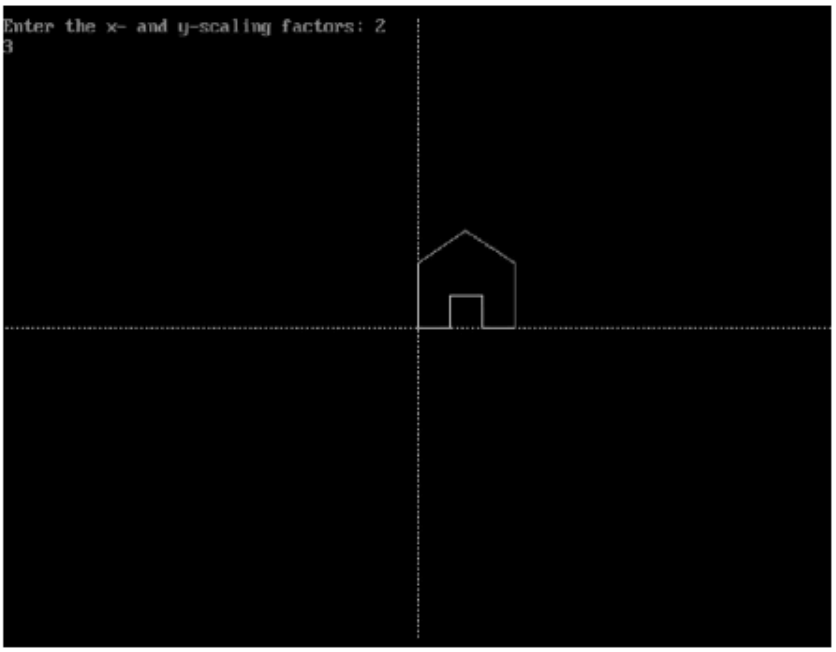
```
                ynew= h[i][0] * sin (angle)+h[i][1] * cos (angle);
                h[i][0]=xnew ; h[i][1] = ynew; } }
void scale (int h[][2], int sx, int sy) {
        int i;
        for (i=0; i<9; i++) {
        h[i][0]  *= sx;
        h[i][1] *= sy; } }
void translate (int h[][2], int dx, int dy) {
        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 () {
```

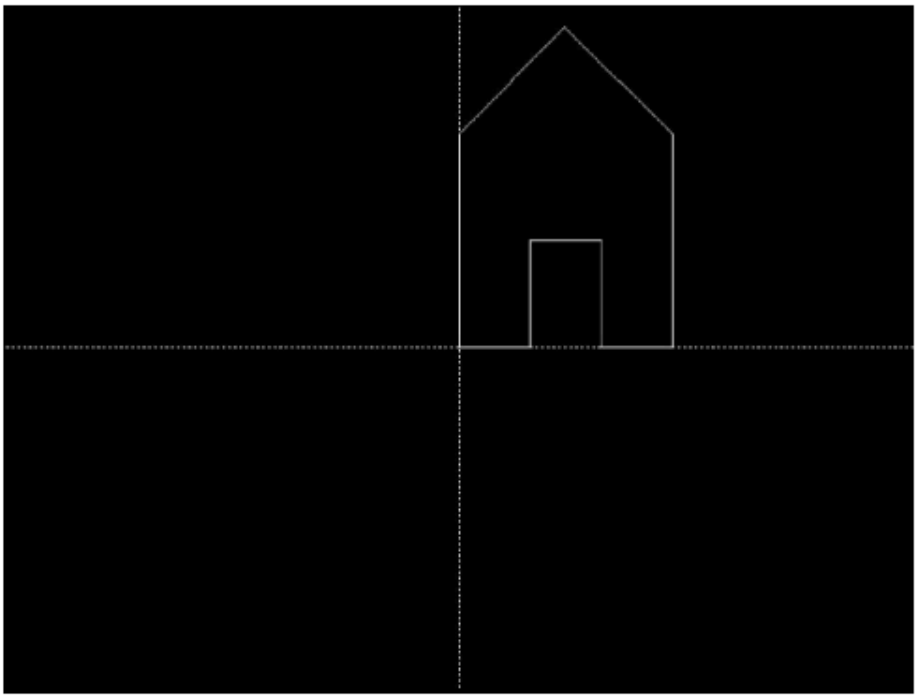
```
int h[9][2],sx,sy,x,y,m,c,choice ;
do {
        clrscr();
        printf("1. Scaling about the origin.\n");
        printf("2. Scaling about an arbitrary point.\n");
        printf("3. Reflection about the line y = mx + c \n");
        printf("4. Exit\n");
        printf("Enter the choice: ");
        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);

                        draw(h);
                        getch();
                        cleardevice();
                        translate(h,-x,-y) ;
                        draw(h);
```

```
                getch();
                cleardevice();
                scale (h, sx, sy);
                draw(h);
                getch();
                translate (h, x, y);
                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 :

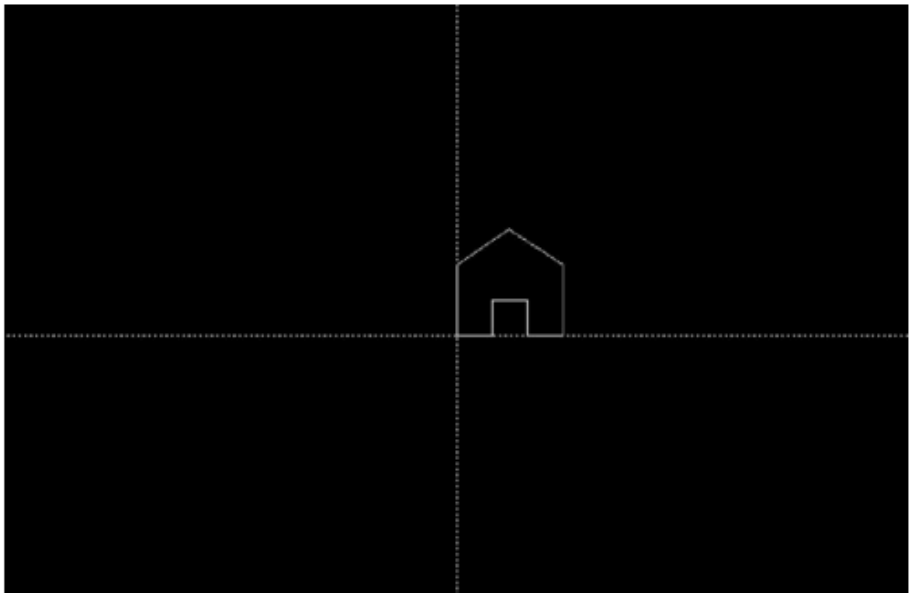
```
1. Scaling about the origin.
2. Scaling about an arbitrary point.
3. Reflection about the line y = mx + c
4. Exit
Enter the choice: 1_
```

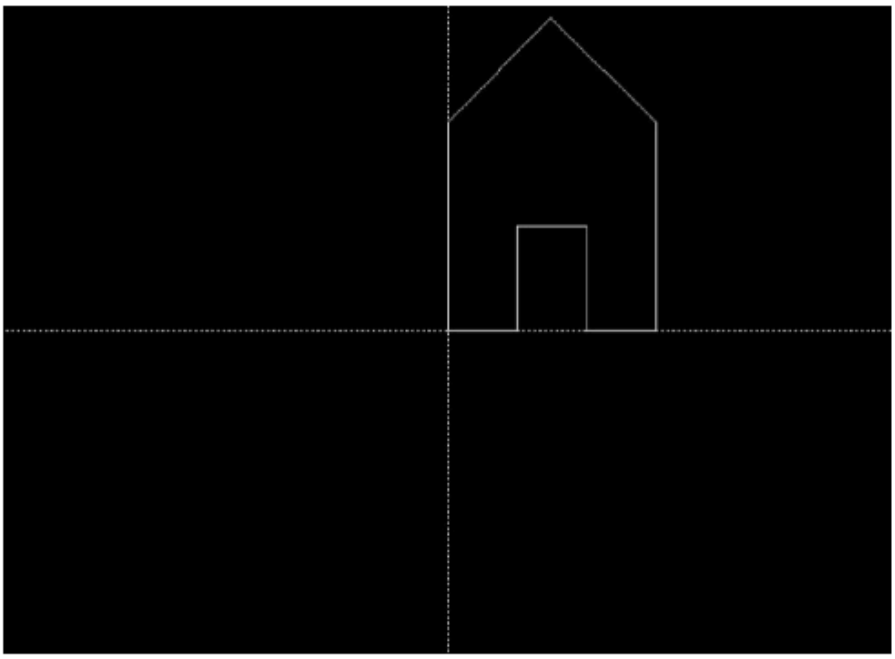

Enter the x- and y-scaling factors: 2
3

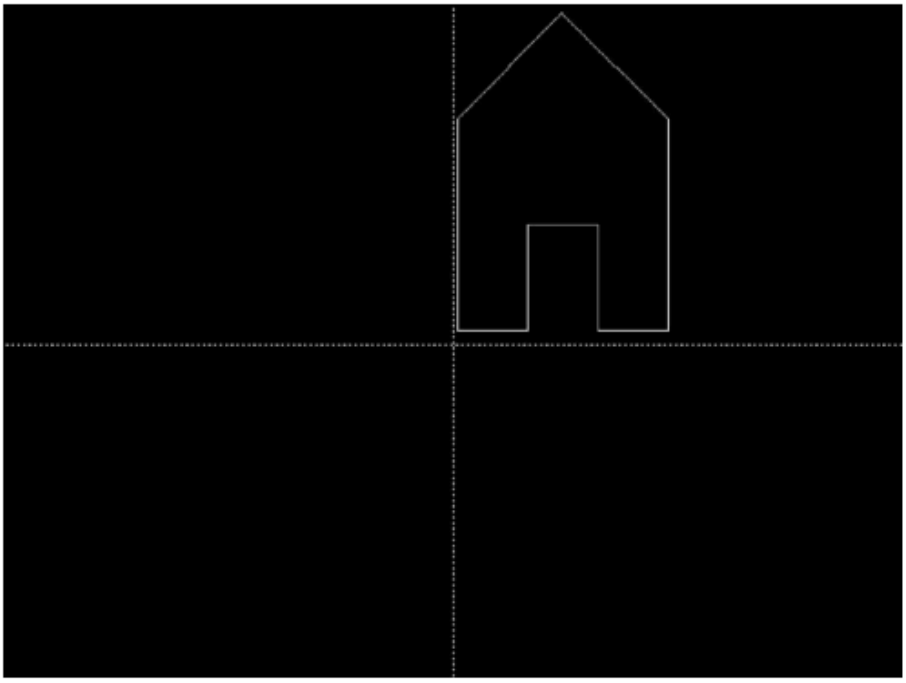


```
1. Scaling about the origin.
2. Scaling about an arbitrary point.
3. Reflection about the line y = mx + c
4. Exit
Enter the choice: 2
```

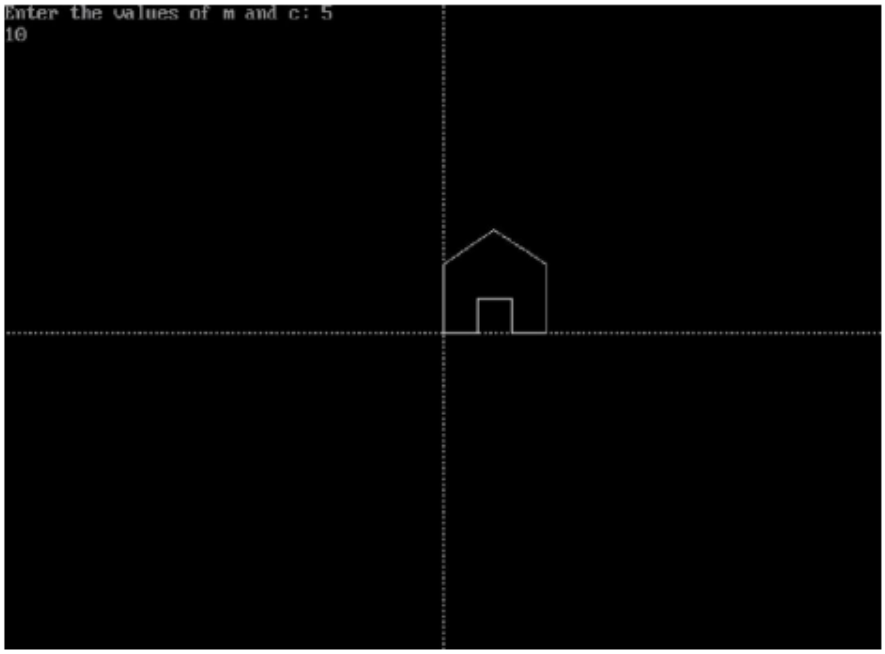
```
Enter the x- and y-scaling factors: 2
3
Enter the x- and y-coordinates of the point: 3
10
```

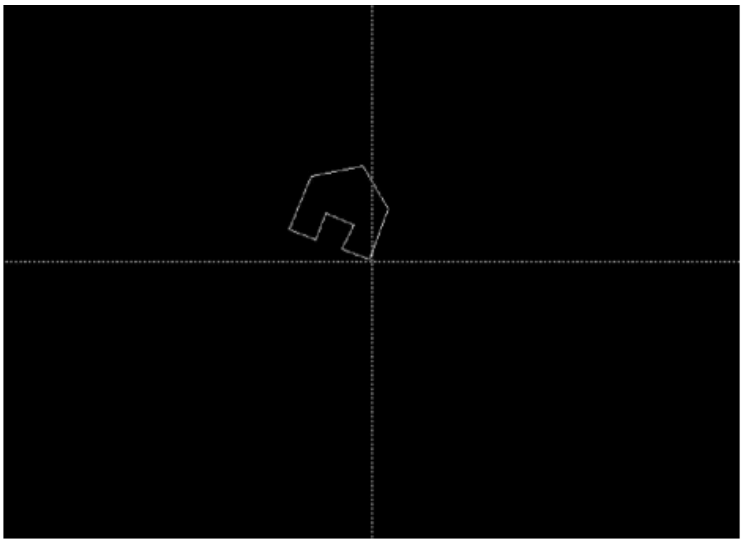






```
1. Scaling about the origin.
2. Scaling about an arbitrary point.
3. Reflection about the line y = mx + c
4. Exit
Enter the choice: 3_
```





```
1. Scaling about the origin.
2. Scaling about an arbitrary point.
3. Reflection about the line y = mx + c
4. Exit
Enter the choice: 4
```

Practical No. 8(A)

Aim : Program to implement Cohen-Sutherland clipping.

Code :

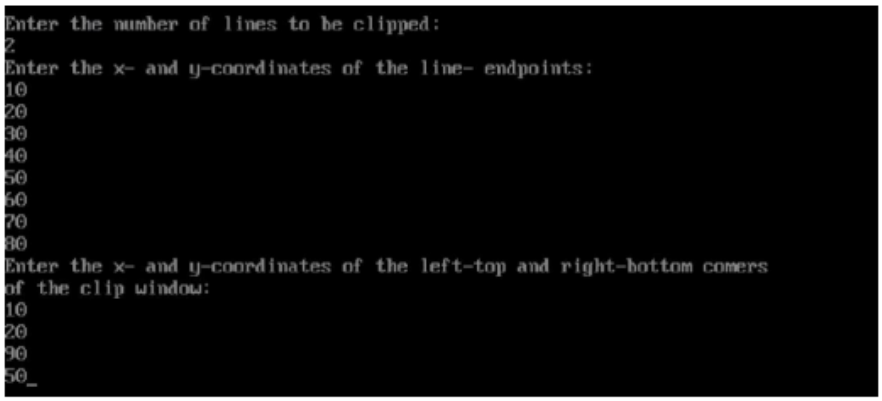
```
 #include <stdio.h>
#include<conio.h>
#include<stdlib.h>
#include <graphics.h>
 #define MAX 20 enum
{
TOP= 0x1, BOTTOM= 0x2, RIGHT = 0x4, LEFT = 0x8 };
enum {FALSE, TRUE};
typedef unsigned int outcode;
outcode compute_outcode(int x, int y,int xmin, int ymin, int xmax, int ymax)
{
  outcode oc = 0;
  if (y > ymax) {
                oc= TOP; }
     else if (y < ymin) {
               oc= BOTTOM; }
        if (x > xmax) {
                     oc = RIGHT; }
         else if (x < xmin) {
                     oc= LEFT; }
          return oc; }
void cohen_sutherland (double x1, double y1, double x2, double y2, double xmin, double ymin,
double xmax, double ymax) {
          int accept;
          int done;
          outcode outcode1, outcode2;
          accept = FALSE;
          done = FALSE;
```

```
outcode1 = compute_outcode (x1, y1, xmin, ymin,
xmax,ymax); outcode2 = compute_outcode (x2, y2, xmin,
ymin, xmax,ymax); do {
        if (outcode1 == 0 && outcode2==0) {
                accept = TRUE;
                done = TRUE; }
        else if (outcode1 & outcode2) {
                done = TRUE; }
        else {
                double x, y;
                int outcode_ex = outcode1 ? outcode1 : outcode2;
                if (outcode_ex & TOP) {
                        x=x1+ (x2-x1) *(ymax-y1)/ (y2-y1);
                        y = ymax; }
                else if (outcode_ex & BOTTOM) {
                        x=x1+ (x2-x1)* (ymin -y1)/ (y2-y1);
                        y = ymin; }
                else if (outcode_ex & RIGHT) {
                        y=y1+ (y2-y1)*(xmax-x1)/(x2-x1);
                        x = xmax; }
                else {
                        y=y1+ (y2-y1)*(xmin-x1)/(x2-x1);
                        x = xmin; }
                if (outcode_ex== outcode1) {
                        x1 = x; y1 = y; outcode1 = compute_outcode(x1, y1, xmin,
                        ymin, xmax, ymax); }

                else {
                        x2 = x;
                        y2 = y;
```

```
                                        outcode2 =compute_outcode (x2, y2, xmin,ymin, xmax, ymax); }
} }
        while (done == FALSE);
        if (accept ==TRUE) {
                line (x1,y1, x2, y2); } }
void main() {
        int n;
        int i, j;
        int ln[MAX][4];
        int clip[4];
        int gd = DETECT, gm;
        clrscr();
        printf ("Enter the number of lines to be clipped:\n");
        scanf("%d", &n) ;
        printf ("Enter the x- and y-coordinates of the line- endpoints: \n");
        for (i=0; i<n; i++) {
                for (j=0; j<4; j++) {
                        scanf("%d", &ln[i][j]); } }
        printf ("Enter the x- and y-coordinates of the left-top and right-");
        printf ("bottom comers\nof the clip window:\n");
        for (i=0; i<4; i++) {
                scanf("%d", &clip[i]); }
        initgraph(&gd, &gm, "C:\\TC\\BGI");
        rectangle(clip[0], clip[1], clip[2], clip[3]) ;
        for (i=0; i<n; i++) {
                line (ln[i][0], ln[i][1], ln[i][2], ln[i][3]); }
        getch () ;
        cleardevice();
        rectangle (clip[0], clip[1], clip[2], clip[3]);
         for (i=0; i<n; i++) {
```

```
                cohen_sutherland (ln[i][0], ln[i][1], ln[i][2], ln[i][3], clip[0], clip[1], clip[2], clip[3]);
                getch(); }
        closegraph () ; }
```

Output :



Practical No. 8(B)

Aim : Program to implement Liang-Barsky Line Clipping Algorithm

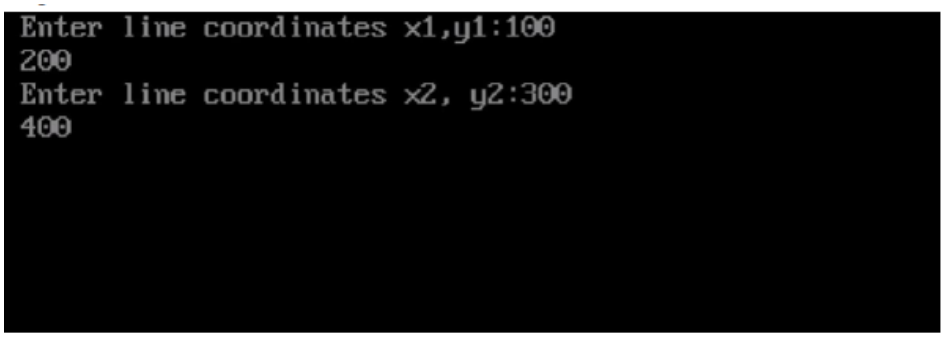
Code:

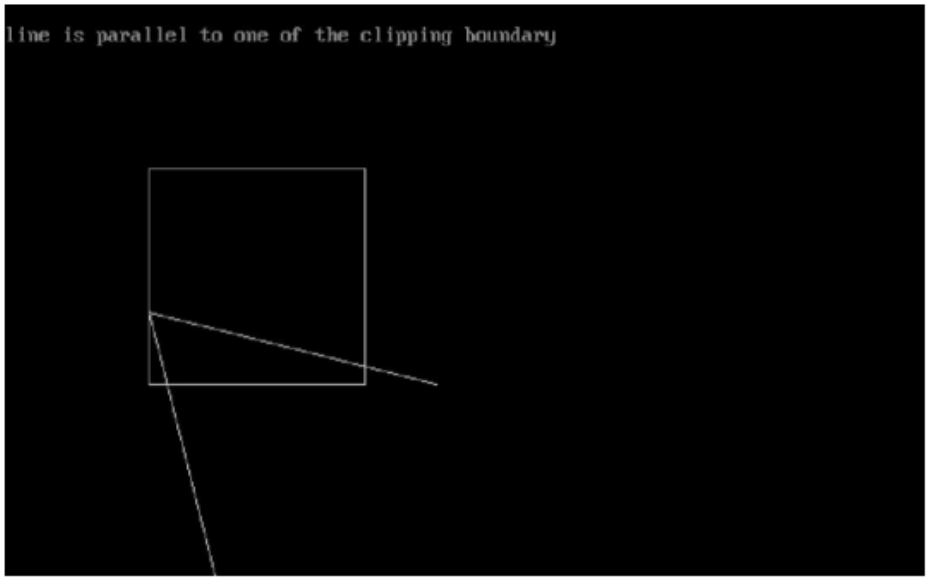
```
#include<stdio.h>
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
int i,gd=DETECT,gm;
int x1,y1,x2,y2,xmin,xmax,ymin,ymax,xx1,xx2,yy1,yy2,dx,dy;
float t1,t2,p[4],q[4],temp;
clrscr();
printf("Enter line coordinates x1,y1:");
scanf("%d %d",&x1,&y1);
printf("Enter line coordinates x2, y2:");
scanf("%d %d",&x2,&y2);
xmin=100;
ymin=100;
xmax=250;
ymax=250;
initgraph(&gd,&gm,"C:\\TC\\BGI");
rectangle(xmin, ymin, xmax, ymax);
dx=x2-x1;
dy=y2-y1;
p[0]=-dx;
p[1]=dx;
p[2]=-dy;
```

```
p[3]=dy;
p[0]=x1-xmin;
p[1]=xmax-x1;
p[2]=y1-ymin;
p[3]=ymax-y1;
for(i=0;i<4;i++) {
        if (p[i]==0) {
                printf("line is parallel to one of the clipping boundary");
                if(q[i]>=0) {
                        if(i<2) {
                                if(y1 <ymin) {
                                        y1=ymin;}
                                if(y2>ymax) {
                                        y2=ymax;}
                                line(x1,y1,x2,y2);}
                        if(i>1) {
                                if(x1<xmin) {
                                        x1=xmin;}
                                if(x2>xmax) {
                                        x2=xmax;}
                                line(x1,y1,x2,y2); } } } }
t1=0;        t2=1;
for(i=0;i<4;i++) {

        temp=q[i]/p[i];
        if(p[i]<0) {
                if(t1<=temp){
                        t1=temp;}
                else {
                        if(t2>temp) {
```

```
                    t2=temp; } } }
        if(t1<t2) {
            xx1 = x1 + t1 * p[1];
            xx2 = x1 + t2 * p[1];
            yy1 = y1 + t1 * p[3];
            yy2 = y1 + 12* p[3];
            line(xx1,yy1,xx2,yy2); }
        delay(9000);
        closegraph(); } }
```

Output :





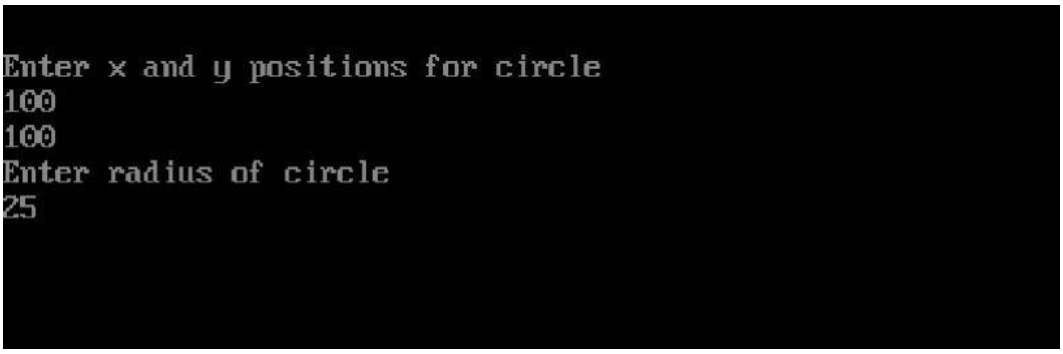
Practical No. 9(A)

Aim : Program to fill a circle using Flood Fill Algorithm.

Code :

```
#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,y+1,oldcolor,newcolor);
                floodFill(x-1,y,oldcolor,newcolor); } }
void 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:\\TC\\BGI");
        circle(x,y,radius); floodFill(x,y,0,15);
        delay(5000);
        closegraph();}
```

Output :



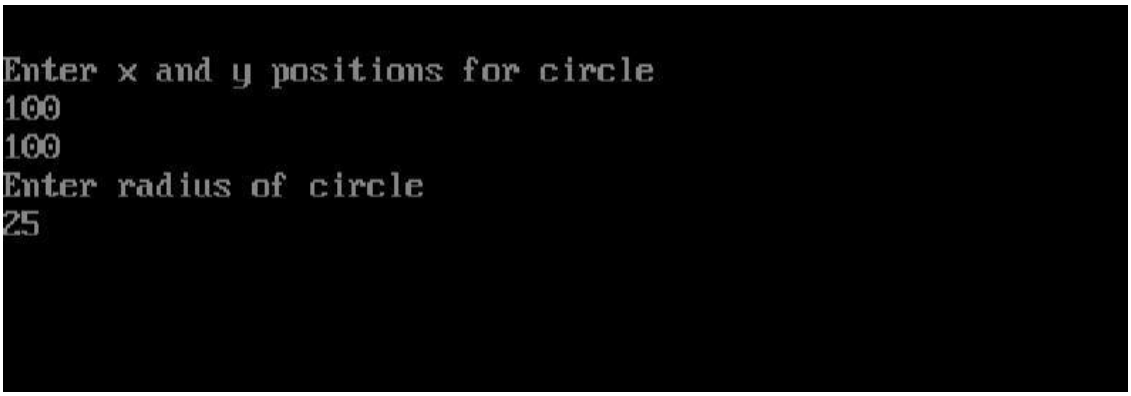
Practical No. 9(B) Aim

: Program to fill a circle using Boundary Fill Algorithm. **Code :**

```
#include<stdio.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); } }
void 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:\\TC\\BGI");
circle(x,y,radius);
boundaryfill(x,y,4,15);
delay(5000);
closegraph(); }
```

Output :

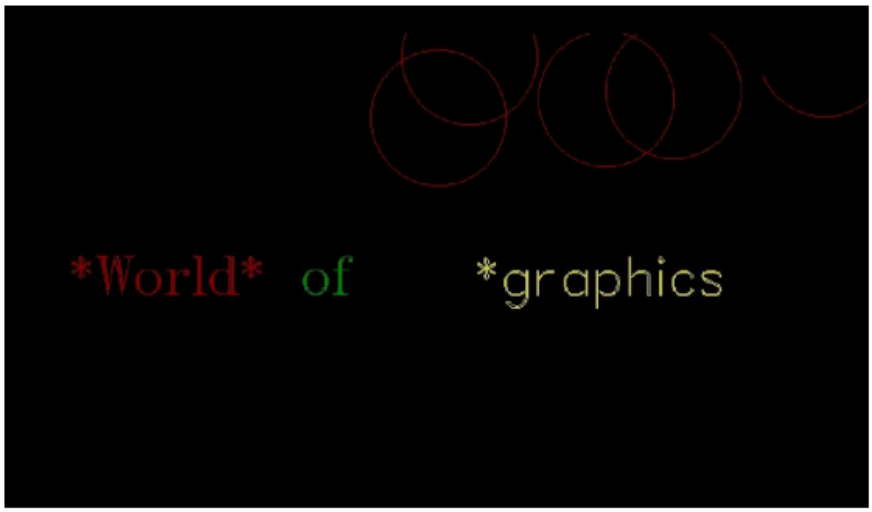


Practical No10(a)

Aim : To develop a simple text screen saver using graphics function.

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main() {
        int gd=DETECT,gm,x=600,i;
        initgraph(&gd, &gm, "C:\\TC\\BGI");
        for(x=0;x<250;x++) {
                x%=250;
                setcolor(random(16));
                circle(random(635),random(70),50);
                circle(random(635),random(70),50);
                circle(random(635),random(70),50);
                circle(random(635),random(70),50);
                circle(random(635),random(70),50);
                clearviewport();
                settextstyle(1,0,5);
                setcolor(RED);
                outtextxy(50,415-2*x,"*World*");
                setcolor(GREEN);
                outtextxy(200,415-2*x," of ");
                setcolor(YELLOW);
                settextstyle (3,0,5);
                outtextxy(350,415-2*x,"*graphics"); }
        getch(); }
```

Output :

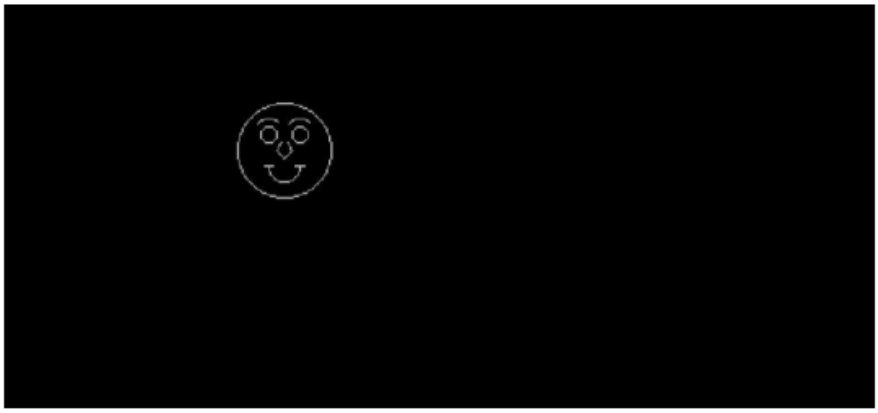


Practical No. 10(b)

Aim : Perform smiling face animation using graphic function.

```
#include <graphics.h>
#include <stdio.h>
#include<conio.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 No. 10(c)

Aim : Draw the moving car on the screen.

```
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <dos.h>
void main() {
        int gd = DETECT, gm, i, maxx, midy ;
        initgraph(&gd, &gm, "C:\\TC\\BGI");
        maxx = getmaxx();
        midy = getmaxy()/2;
        for (i=0; i < maxx-150; i=i+5) {
                cleardevice();
                setcolor(WHITE);
                line(0, midy + 37, maxx, midy + 37);
                setcolor(YELLOW);
                setfillstyle(SOLID_FILL, RED);
                line(i, midy + 23, i, midy);
                line(i, midy, 40+ i, midy - 20);
                line(40+ i, midy - 20, 80+ i, midy - 20);
                line (80+i, midy - 20, 100 + i, midy);
                line(100+i, midy, 120+ i, midy);
                line(120+i, midy, 120+ i, midy + 23);
                line(0+ i, midy + 23, 18+ i, midy + 23);
                arc(30+ i, midy + 23, 0, 180, 12);
                line(42+ i, midy + 23, 78+ i, midy + 23);
                arc(90+ i, midy + 23, 0, 180, 12);
                 line(102+i, midy + 23, 120 + i, midy + 23);
                 line(28+ i, midy, 43 + i, midy - 15);
                 line(43+ i, midy - 15, 57 + i, midy - 15);
                 line(57+i, midy - 15, 57 + i, midy);
                 line(57+i, midy, 28 + i, midy);
                 line(62+ i, midy - 15, 77+ i, midy - 15);
                 line(77+ i, midy - 15, 92+ i, midy);
                 line(92+i, midy, 62+ i, midy);
```

```
        line(62+ i, midy, 62 + i, midy - 15);
        floodfill (5+i,midy + 22, YELLOW);
        setcolor(BLUE);
        setfillstyle (SOLID_FILL, DARKGRAY);
        circle(30+i, midy + 25, 9);
        circle(90+ i, midy + 25, 9);
        floodfill (30+ i, midy + 25, BLUE);
        floodfill(90+ i, midy + 25, BLUE);
        delay(100); }
    getch() ;
    closegraph (); }
```

Output :