Practical 1a

Aim: Study and enlist the basic functions used for graphics in C / C++ / Python language. Give

NAME	FUNCTION	SYNTAX
1.arc	arc draws a circular arc.	void far arc(int x,int y,int stangle,int endangle,int radius);
2.circle	circle draws a circle.	void far circle(int x,int y,int radius);
3.bar	draws a bar.	void far bar(int left,int top,int right,int bottom);
4.closegraph	shutdown the graphic system.	void far closegraph(void);
5.ellipse	ellipse draws an elliptical arc.	void far ellipse(int x,int y,int stangle,int endangle,int xradius,int yradius);
6.floodfill	floodfills a bound region.	void far floodfill(int x,int y,int radius);
7.getbkcolor	getbkcolor returns the current back ground color.	int far getbkcolor(void);
8.getgraphmode	getgraphmode returns the current graphic mode.	int far getgraphmode(void);
9.getmaxcolor	returns the maximum color value.	int far getmaxcolor(void);
10.getmaxx	returns maximum x screen coordinate.	int far gatmaxx(void);
11.getmaxy	returns maximum yscreen coordinate.	int far getmaxy(void);
12.gety	returns the current positions y coordinate.	int far gety(viod);

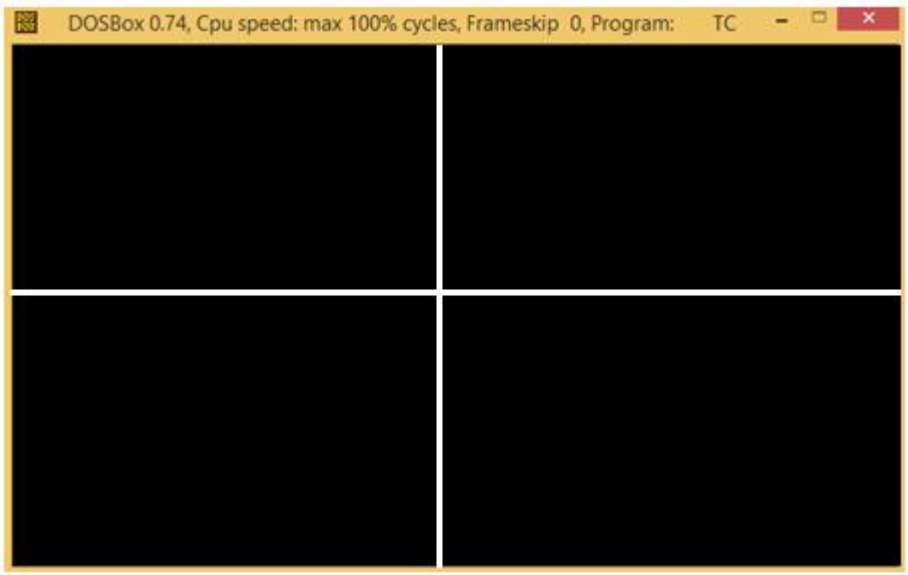
13.getx	returns the current positions xcoordinate.	int far getx(void);
14.detectgraph	determaines graphic driver and mode to use by checking the hardware.	void far detectgraph(int far *graphdriver,int far *graphmode);
15.fillellipse	fillellipse draws and fill an ellipse.	void far fillellipse(int x,int y,int xradius,int yradius);
16.getarccoords	gets coordinate of the last call to arc.	void far getarccoords(struct arccoords type far *arccoords);
17.getcolor	returns the current drawing color.	int far getcolor(void);
18.getfillpattern	copies a userefined fill pattern into memory.	void far getfillpattern(char far *pattern);
19.getmaxmode	returns maximum graphics mode number for current driver.	int far getmaxmode(void);
20.drawpoly	draws the outline of a polygon.	void far drawpoly(int numpoints,int far *polypoints);
21.fillpoly	fillpoly draws and fills a polygon.	void far fillpoly(int numpoints,int *polypoints);
22.clearviewpoint	clear the current view port.	void far clearviewport(void);
23.getpixel	getpixel gets the color of a specified pixel.	unsigned far getpixel(int x,int y);
24.grapherrormsg	returns a pointer to an error message string.	char *far grapherrormsg(int errorcode);
25.lineto	draws a line from th current position cp to	void far lineto(int x,int y);

	(x,y).	
26.line	draws a line between two specified points.	void far line(int x1,int y1,int x2,int y2);
27.initgraph	initialize the graphic system.	void far initgraph(int far *graphdrive,int far *graphmode,int far *pathtodrive);
28.rectangle	draws a rectangle.	void far rectangle(int left,int top,int right,int bottom);
29.putpixel	plots a pixel at a specified point.	void far putpixel(int x,int y,int color);
30.imagesize	returns the number of bytes required to store a bit image.	Unsigned far imagesize(int left,int top,int right,int bottom);
31.moveto	Moves the cp to (x,y).	void far moveto(int x,int y);
32.setcolor	sets the current drawing color.	void far setcolor(int color);
33.setgraphmode	sets the system to graphics mode,clear the screen.	void far setgraphmode(int mode);
34.textwidth	returns the width of string in pixels.	int far textwidth(char far *textstring);
35.textheight	returns the height of string in pixels.	int far textheight(char far *textstring);

Practical 1 b

Aim: Draw a co-ordinate axis at the center of the screen

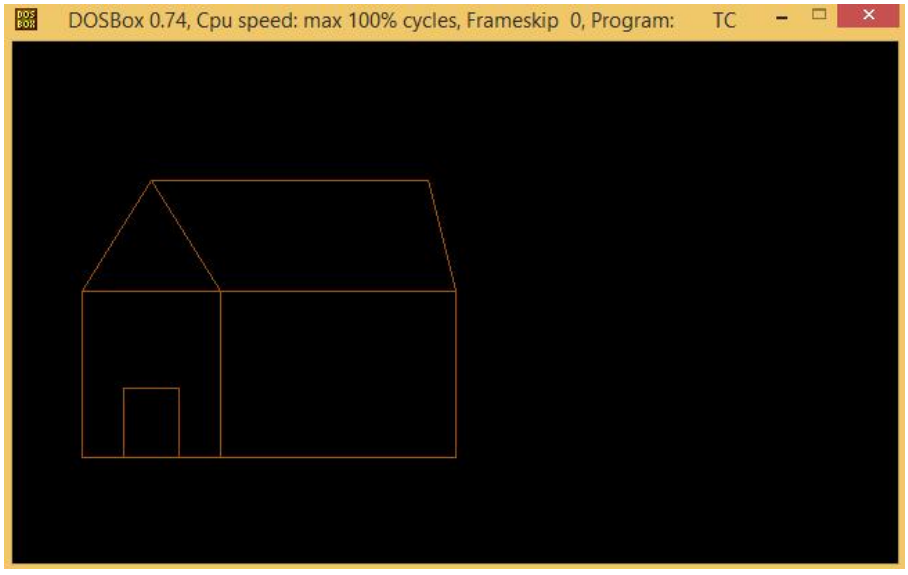
```
#include<graphics.h>
#include<conio.h>

void main()
{
   int gd = DETECT,gm=100,xcen,ycen;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   xcen=getmaxx()/2;
   ycen=getmaxy()/2;
line(xcen,0,xcen,getmaxy());
line(0,ycen,getmaxx(),ycen);

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



Practical 2a

Aim: Draw a simple hut on the screen.

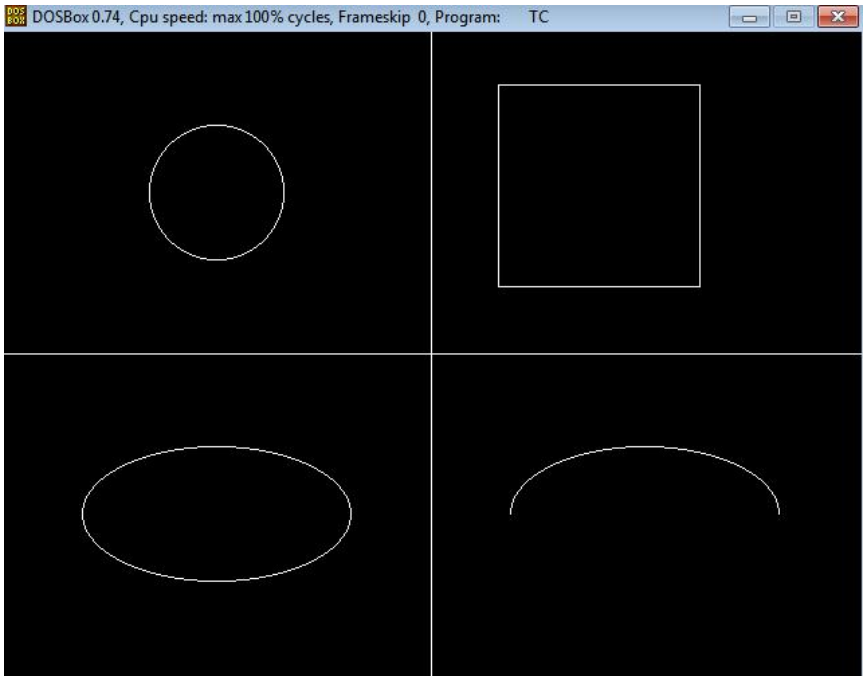
```
#include<graphics.h>
#include<conio.h>

void main()
{
clrscr();
  int gd = DETECT,gm=100;
  initgraph(&gd, &gm, "C:\\TC\\BGI");

setcolor(10);
rectangle(50,180,150,300);
rectangle(150,180,320,300);
rectangle(80,250,120,300);
line(100,100,50,180);
line(100,100,150,180);
line(100,100,300,100);
line(300,100,320,180);
getch();
closegraph();
}
```



Practical 2b

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

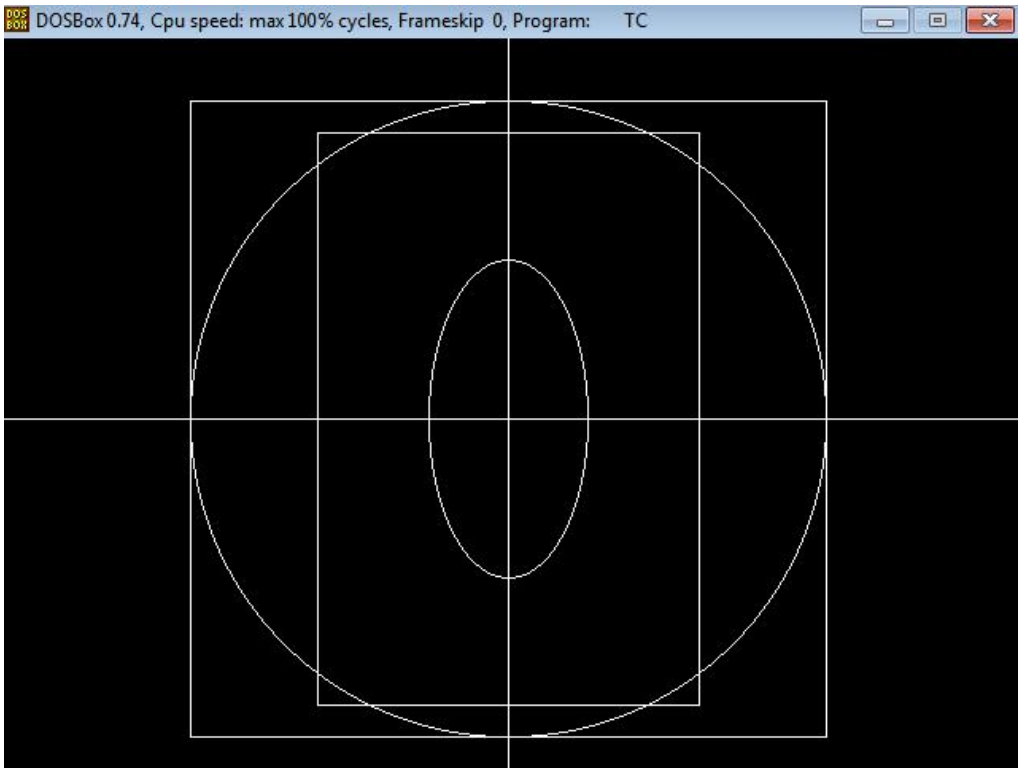
```
#include<graphics.h>
#include<conio.h>

void main()
{
   int gd = DETECT,gm=100,xcen,ycen,i;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   xcen=getmaxx()/2;
   ycen=getmaxy()/2;
line(xcen,0,xcen,getmaxy());
line(0,ycen,getmaxx(),ycen);

circle(xcen/2,ycen/2,50);
rectangle(xcen+50,ycen-200,xcen+200,ycen-50);
ellipse(xcen/2,ycen+ycen/2,0,360,100,50);
ellipse(xcen+xcen/2,ycen+ycen/2,0,180,100,50);


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



Practical 3a

Aim: Draw the following basic shapes in the center of the screen : i. Circle ii. Rectangle iii. Square iv.Ellipse v. Line

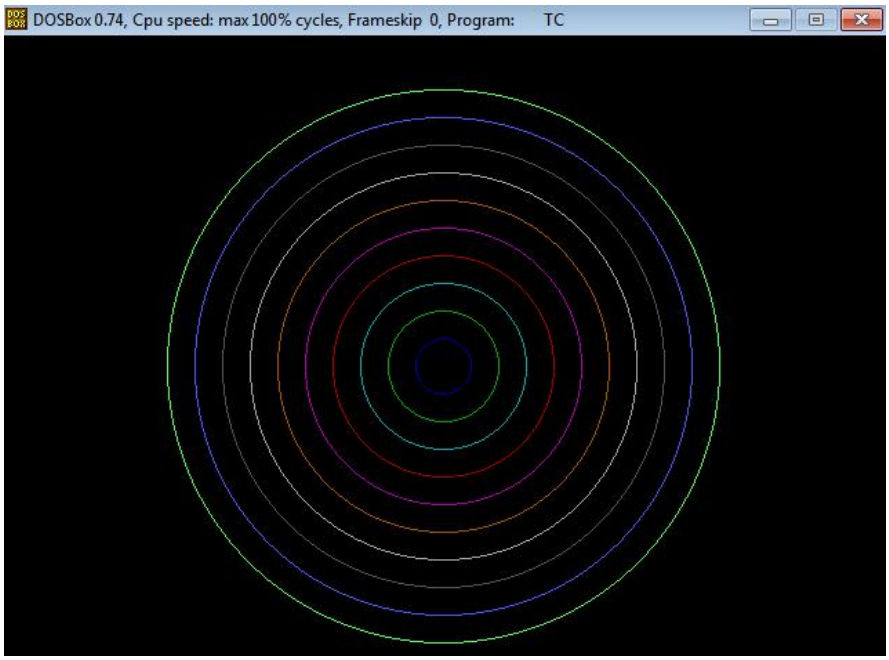
```
#include<graphics.h>
#include<conio.h>

void main()
{
   int gd = DETECT,gm=100,xcen,ycen,i;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   xcen=getmaxx()/2;
   ycen=getmaxy()/2;
line(xcen,0,xcen,getmaxy());
line(0,ycen,getmaxx(),ycen);

circle(xcen,ycen,200);
rectangle(xcen-200,ycen-200,xcen+200,ycen+200);//square
rectangle(xcen-120,ycen-180,xcen+120,ycen+180);
ellipse(xcen,ycen,0,360,50,100);


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



Practical 3b

Aim: Practical on Concentric Circles

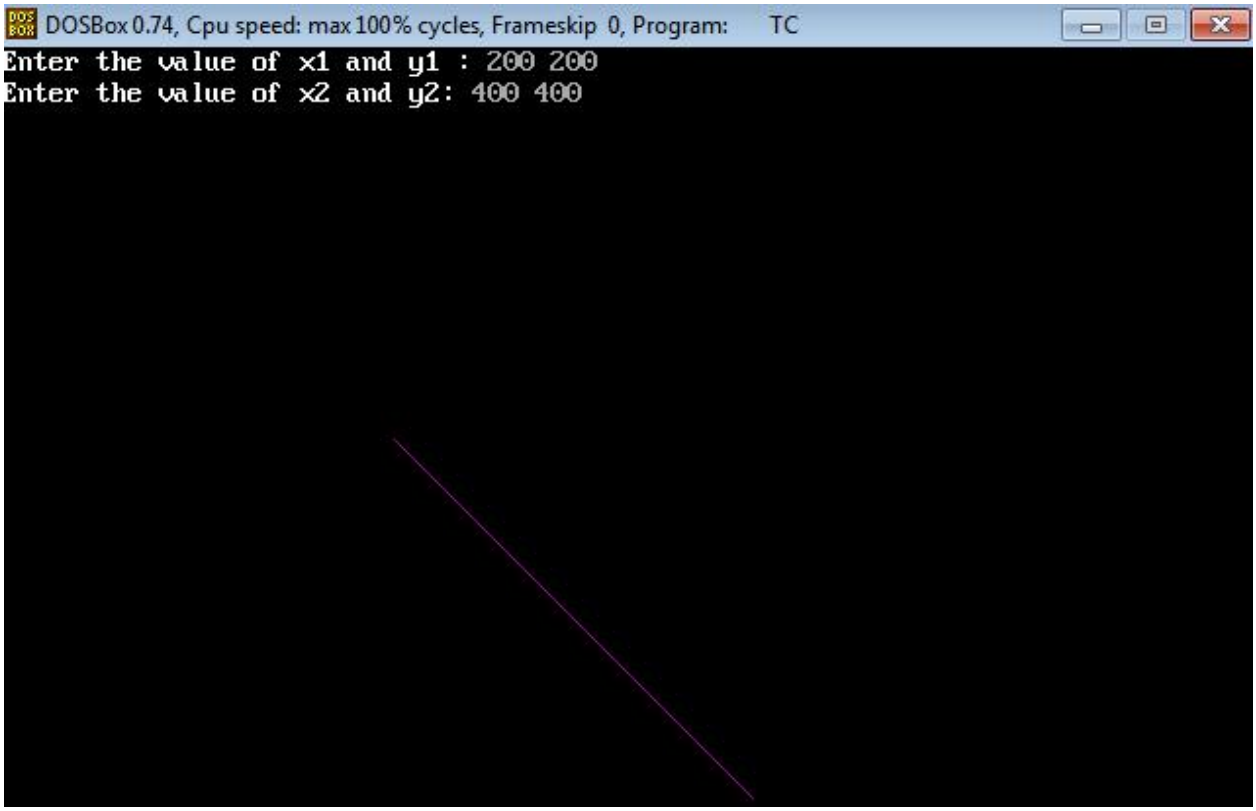
```
#include<graphics.h>
#include<conio.h>

void main()
{
   int gd = DETECT,gm=100,xcen,ycen,i,color=1;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   xcen=getmaxx()/2;
   ycen=getmaxy()/2;
for(i=20;i<=200;i+=20)
{


setcolor(color++);
circle(xcen,ycen,i);


}

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



Practical 4a

Aim: Program for DDA Line Drawing Algorithm in C++

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

void main( )
{
	float x,y,x1,y1,x2,y2,dx,dy,step;
	  int gd = DETECT, gm;
		    initgraph(&gd, &gm, "C:\\TC\\BGI");


	cout<<"Enter the value of x1 and y1 : ";
	cin>>x1>>y1;
	cout<<"Enter the value of x2 and y2: ";
	cin>>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;

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

	getch();
closegraph();

}
```

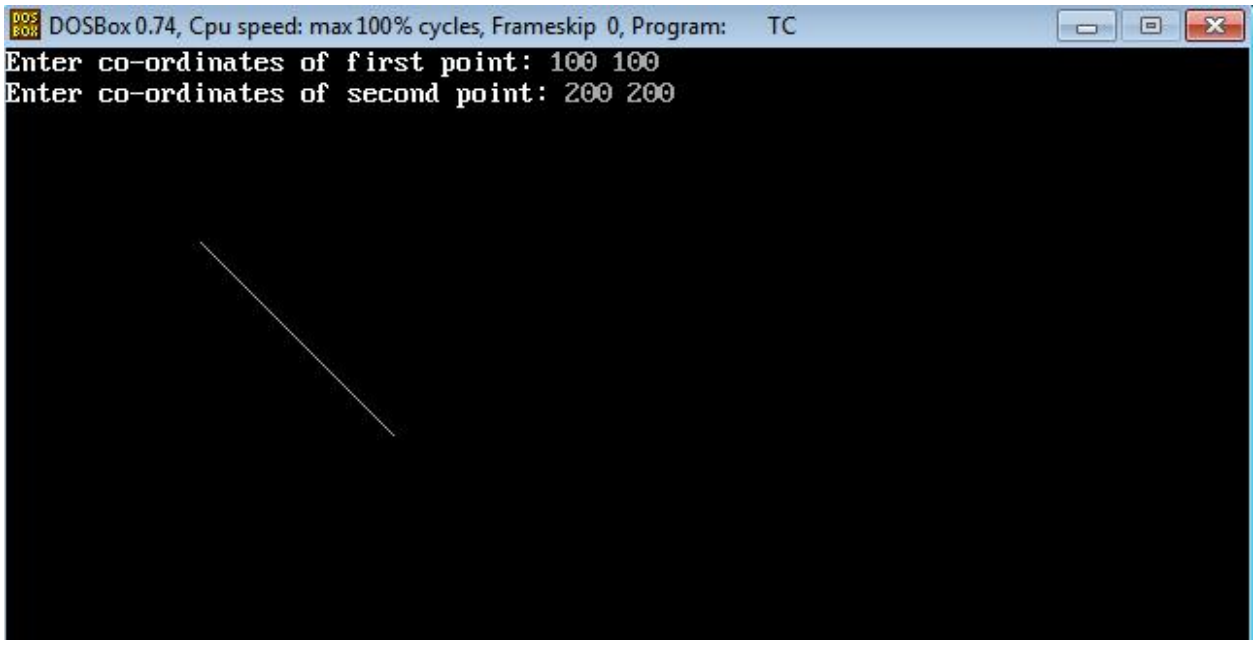


Practical 4b

Aim: Program for Bresenham's Line Drawing Algorithm in C++

```
#include<stdio.h>
#include<conio.h>
#include<iostream.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);
```

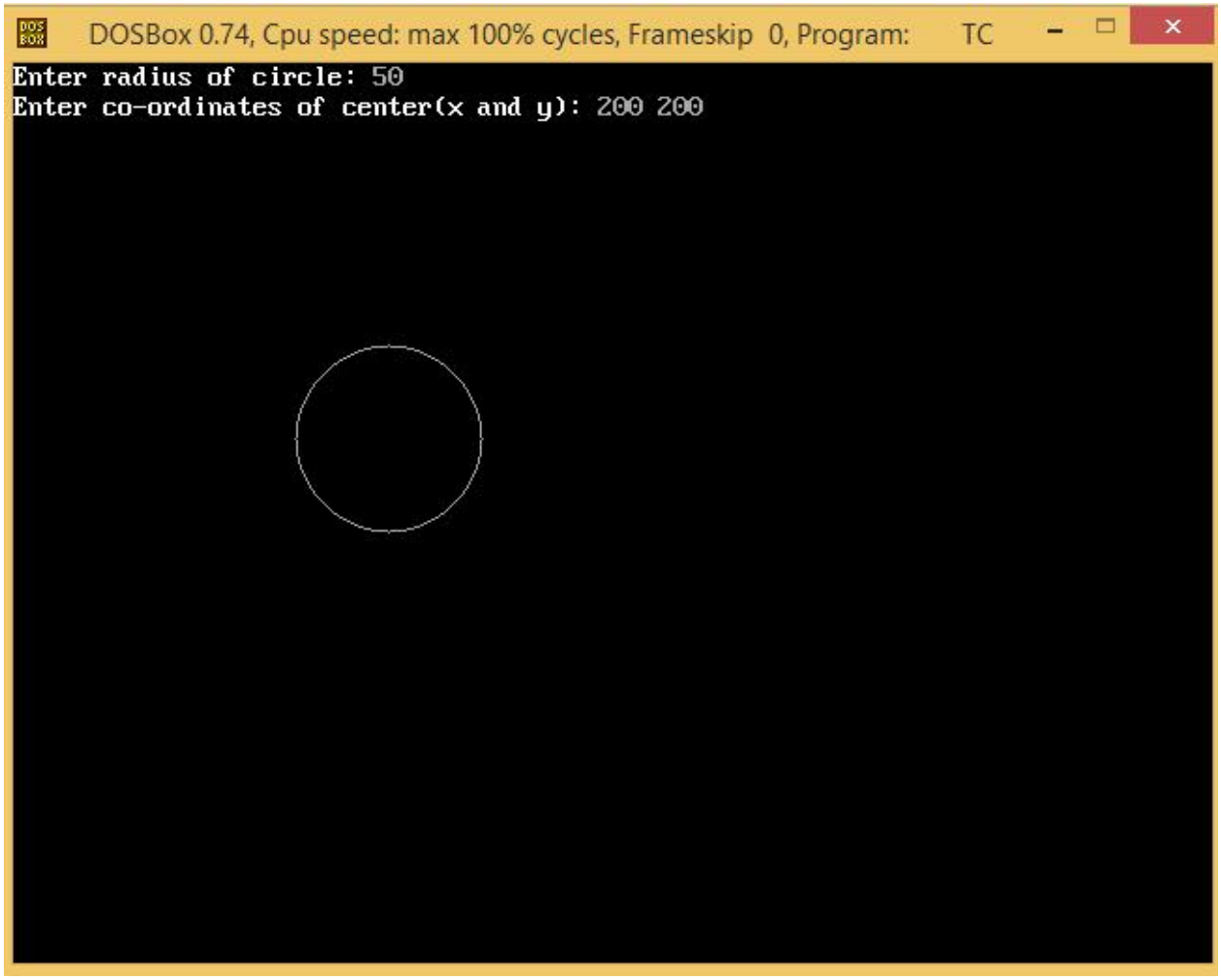
```
                        y=y+1;
                        p=p+2*dy-2*dx;
                }
                else
                {
                        putpixel(x,y,7);
                        p=p+2*dy;
                }
                x=x+1;
        }
}

void main()
{
         int gd = DETECT, gm, error, x0, y0, x1, y1;;
                   initgraph(&gd, &gm, "C:\\TC\\BGI");

        cout<<"Enter co-ordinates of first point: ";
        cin>>x0>>y0;

        cout<<"Enter co-ordinates of second point: ";
        cin>>x1>>y1;
        drawline(x0, y0, x1, y1);

        getch();
closegraph();

}
```



Practical 5a
Aim: Program for Midpoint Circle Algorithm in C++

```
#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;
          }
   }
}

void main()
{

int gd = DETECT,gm=100, error, x, y, r;;
  initgraph(&gd, &gm, "C:\\TC\\BGI");

          cout<<"Enter radius of circle: ";
          cin>>r;
```

```
        cout<<"Enter co-ordinates of center(x and y): ";
        cin>>x>>y;
        drawcircle(x, y, r);

        getch();
}
```

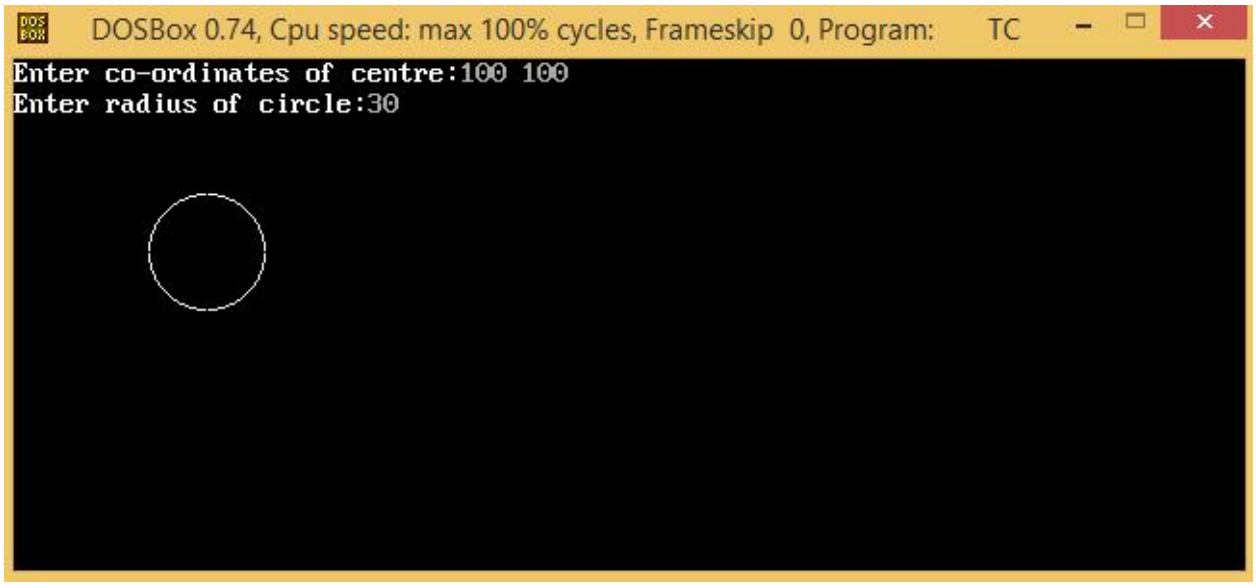


Practical 5b

Aim: Write a program to implement Mid-Point Ellipse Generation Algorithm.

```
#include<graphics.h>
#include<iostream.h>
#include<conio.h>
void main()
{
int xc,yc,x,y,d,r;
int gdriver = DETECT, gmode;
initgraph(&gdriver,&gmode,"C:\\TC\\BGI");
cout<<"Enter co-ordinates of centre:";
cin>>xc>>yc;
cout<<"Enter radius of circle:";
cin>>r;
x = 0;
y = r;
d = 3-2*r;
do{
if(d<0){
d += 4*x +6;
x++;
}
else{
d += 4*x-4*y +10;
```

```
x++;
y--;
}
putpixel(xc+x,yc+y,WHITE);
putpixel(xc+y,yc+x,WHITE);
putpixel(xc+x,yc-y,WHITE);
putpixel(xc+y,yc-x,WHITE);
putpixel(xc-x,yc-y,WHITE);
putpixel(xc-y,yc-x,WHITE);
putpixel(xc-x,yc+y,WHITE);
putpixel(xc-y,yc+x,WHITE);
}
while(x<y);
getch();
closegraph();
}
```

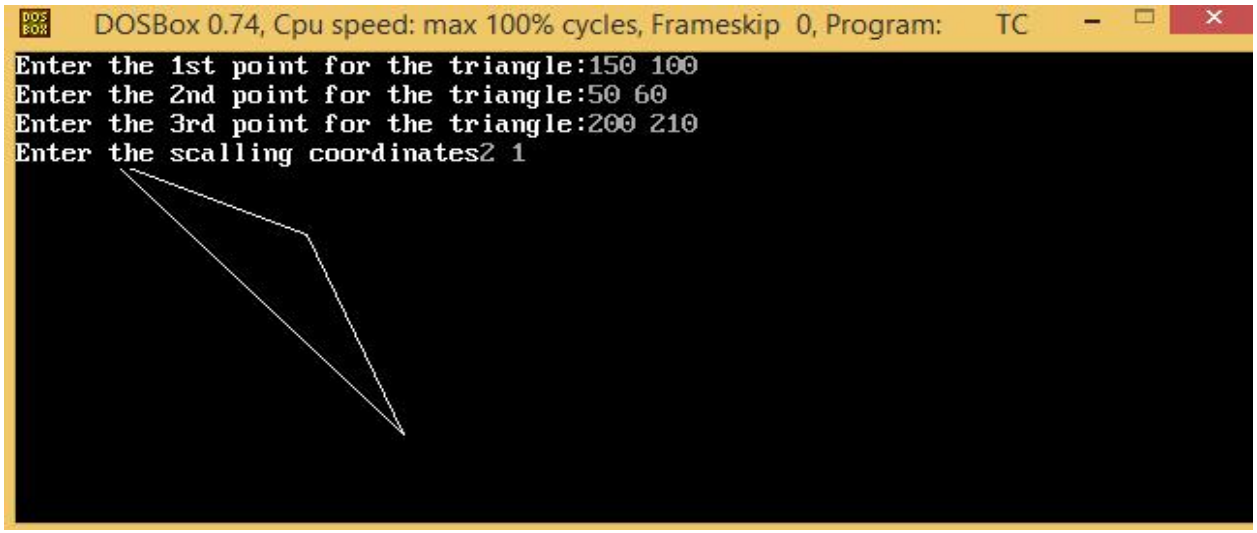


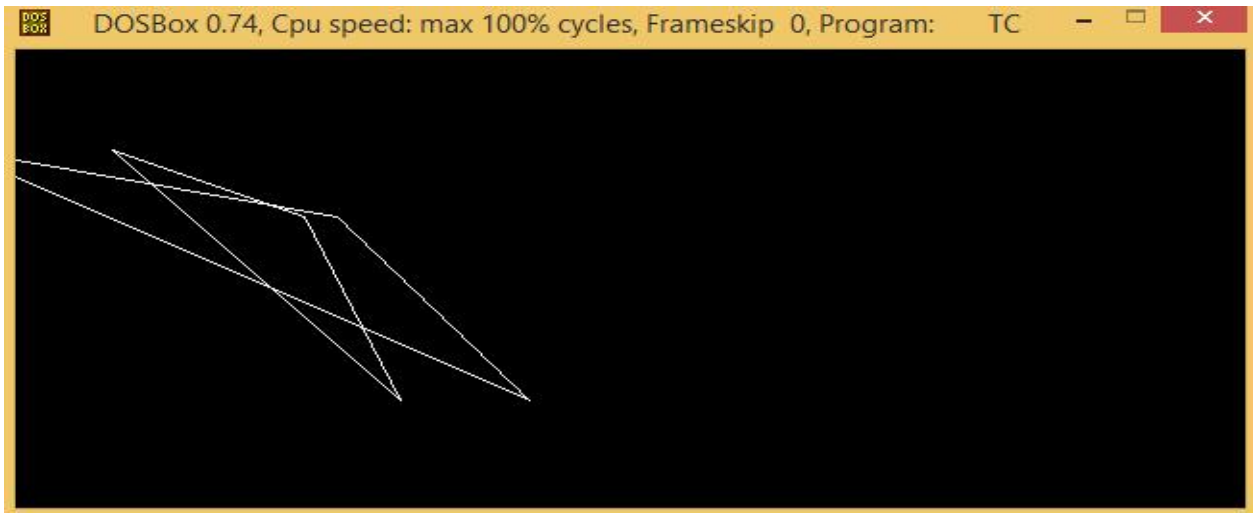
Practical 6a

Aim: Program for 2D Scaling of a triangle

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<math.h>
 int x1,y1,x2,y2,x3,y3,mx,my;
void draw();
void scale();
 void main()
{
  int gd=DETECT,gm;
  int c;
  initgraph(&gd,&gm,"c:\\tc\\bgi");
  printf("Enter the 1st point for the triangle:");
  scanf("%d%d",&x1,&y1);
  printf("Enter the 2nd point for the triangle:");
  scanf("%d%d",&x2,&y2);
  printf("Enter the 3rd point for the triangle:");
  scanf("%d%d",&x3,&y3);
  draw();
  scale();
}
 void draw()
{
  line(x1,y1,x2,y2);
  line(x2,y2,x3,y3);
  line(x3,y3,x1,y1);
}
void scale()
{
  int x,y,a1,a2,a3,b1,b2,b3;
  int mx,my;
  printf("Enter the scalling coordinates");
  scanf("%d%d",&x,&y);
  mx=(x1+x2+x3)/3;
  my=(y1+y2+y3)/3;
  cleardevice();
  a1=mx+(x1-mx)*x;
  b1=my+(y1-my)*y;
```

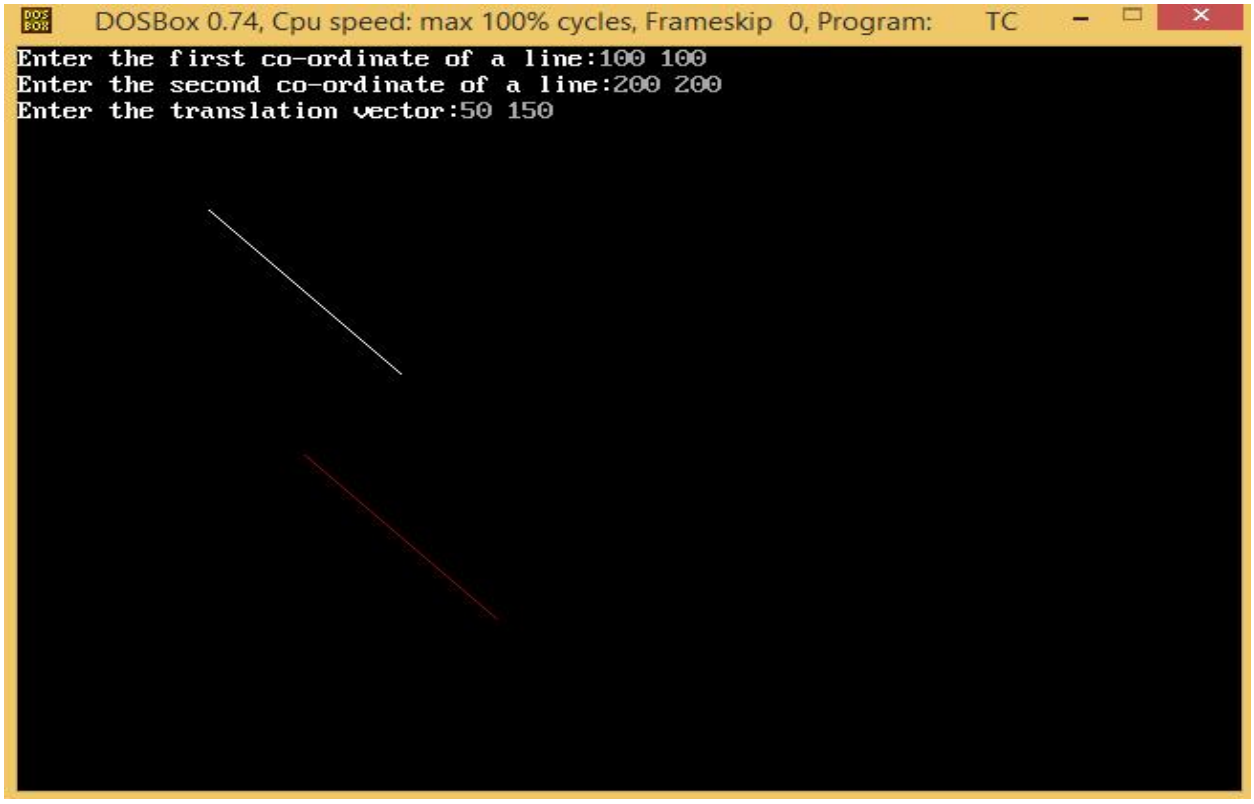
```
  a2=mx+(x2-mx)*x;
   b2=my+(y2-my)*y;
   a3=mx+(x3-mx)*x;
  b3=my+(y3-my)*y;
  line(a1,b1,a2,b2);
  line(a2,b2,a3,b3);
  line(a3,b3,a1,b1);
  draw();
  getch();

}
```





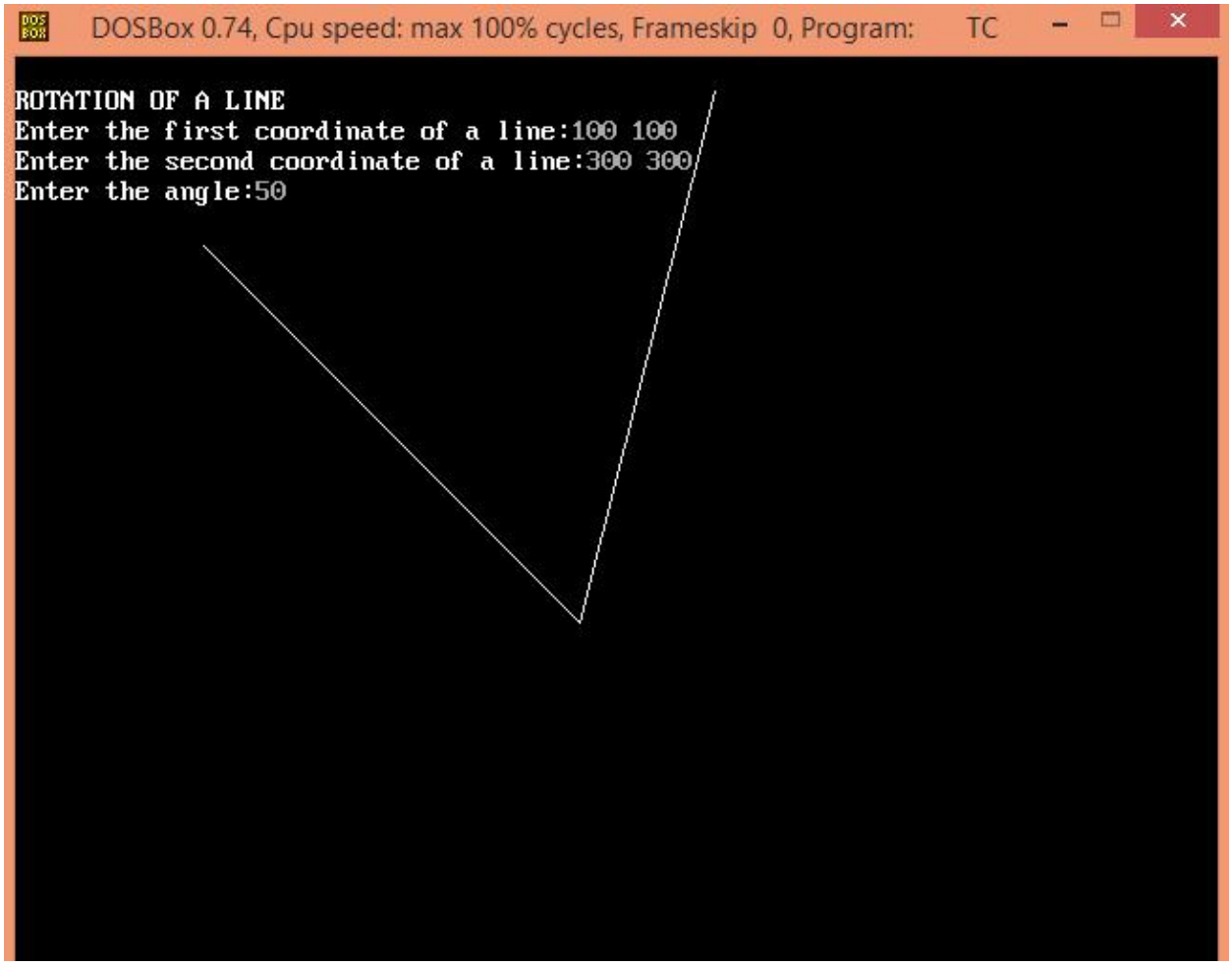
Practical 6b
Aim: Practical for TRANSLATION

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

void main()
{
        int gd=DETECT,gm,x1,x2,y1,y2,tx,ty;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        cout<<"Enter the first co-ordinate of a line:";
        cin>>x1>>y1;
        cout<<"Enter the second co-ordinate of a line:";
        cin>>x2>>y2;
        line(x1,y1,x2,y2);
        cout<<"Enter the translation vector:";
        cin>>tx>>ty;
        setcolor(RED);
        x1=x1+tx;
        y1=y1+ty;
        x2=x2+tx;
        y2=y2+ty;
        line(x1,y1,x2,y2);
        getch();
        closegraph();
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter the first co-ordinate of a line:100 100
Enter the second co-ordinate of a line:200 200
Enter the translation vector:50 150

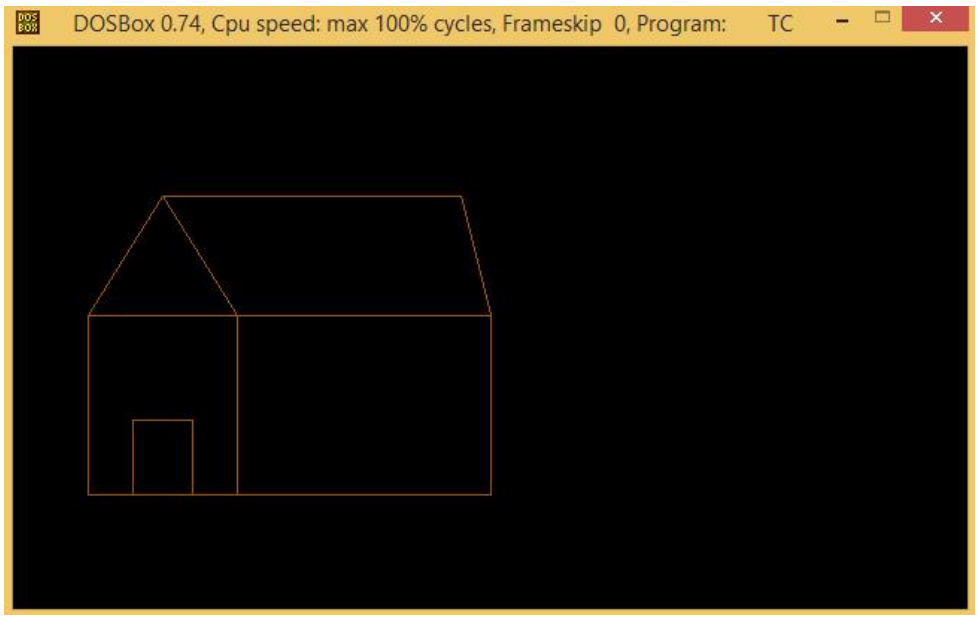
Practical 7a
Aim: Practical for ROTATION OF A LINE

```
#include<math.h>
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        int gd=DETECT,gm,x1,x2,y1,y2,x4,y4;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        float angle=0,ang;
        cout<<"\nROTATION OF A LINE\n";
        cout<<"Enter the first coordinate of a line:";
        cin>>x1>>y1;
        cout<<"Enter the second coordinate of a line:";
        cin>>x2>>y2;
        line(x1,y1,x2,y2);
        cout<<"Enter the angle:";
        cin>>ang;
        angle=(ang*3.14)/180;
        setcolor(RED);
        x4=x2-(((x2-x1)*cos(angle))-((y2-y1)+sin(angle)));
        y4=y2-(((x2-x1)*sin(angle))+((y2-y1)*cos(angle)));
        line(x2,y2,x4,y4);
        getch();
        closegraph();
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
ROTATION OF A LINE
Enter the first coordinate of a line:100 100
Enter the second coordinate of a line:300 300
Enter the angle:50

Practical 7b
Aim: Program to make a house

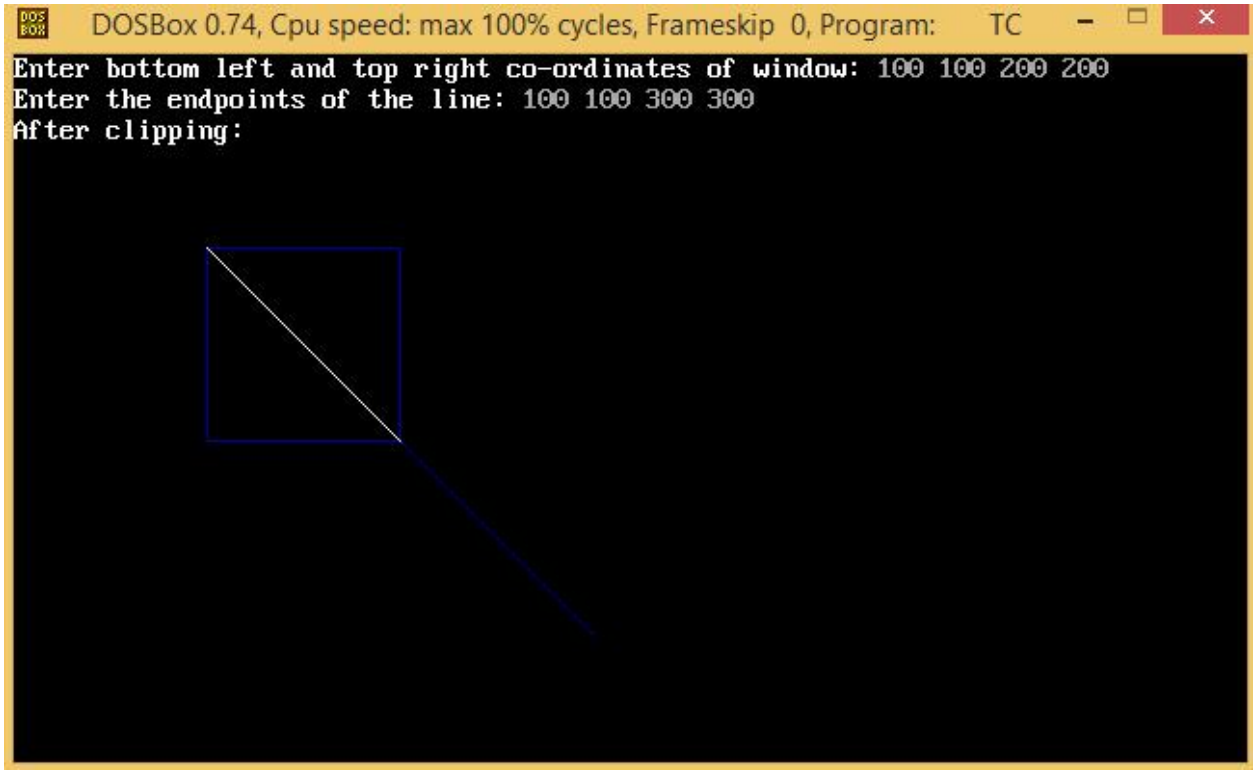
```
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
void main()
{
clrscr();
int gd = DETECT,gm;
initgraph(&gd,&gm,"C:\\TC\\BGI");
setcolor(6);
rectangle(50,180,150,300);
rectangle(150,180,320,300);
rectangle(80,250,120,300);
line(100,100,50,180);
line(100,100,150,180);
line(100,100,300,100);
line(300,100,320,180);
getch();
closegraph();
}
```



Practical 8a
Aim: Program to implement Cohen-Sutherland Line Clipping Algorithm in C++

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

static int LEFT=1,RIGHT=2,BOTTOM=4,TOP=8,xl,yl,xh,yh;

int getcode(int x,int y){
	int code = 0;
	//Perform Bitwise OR to get outcode
	if(y > yh) code |=TOP;
	if(y < yl) code |=BOTTOM;
	if(x < xl) code |=LEFT;
	if(x > xh) code |=RIGHT;
	return code;
}

void main()
{
	int gdriver = DETECT,gmode=100;
	initgraph(&gdriver,&gmode,"C:\\TC\\BGI");
	setcolor(BLUE);
	cout<<"Enter bottom left and top right co-ordinates of window: ";
	cin>>xl>>yl>>xh>>yh;
	rectangle(xl,yl,xh,yh);
	int x1,y1,x2,y2;
	cout<<"Enter the endpoints of the line: ";
	cin>>x1>>y1>>x2>>y2;
	line(x1,y1,x2,y2);
	getch();

	int outcode1=getcode(x1,y1), outcode2=getcode(x2,y2);
	int accept = 0;   //decides if line is to be drawn
	while(1){
		float m =(float)(y2-y1)/(x2-x1);
		//Both points inside. Accept line
		if(outcode1==0 && outcode2==0){
			accept = 1;
			break;
		}
		//AND of both codes != 0.Line is outside. Reject line
		else if((outcode1 & outcode2)!=0){
			break;
		}else{
			int x,y;
```

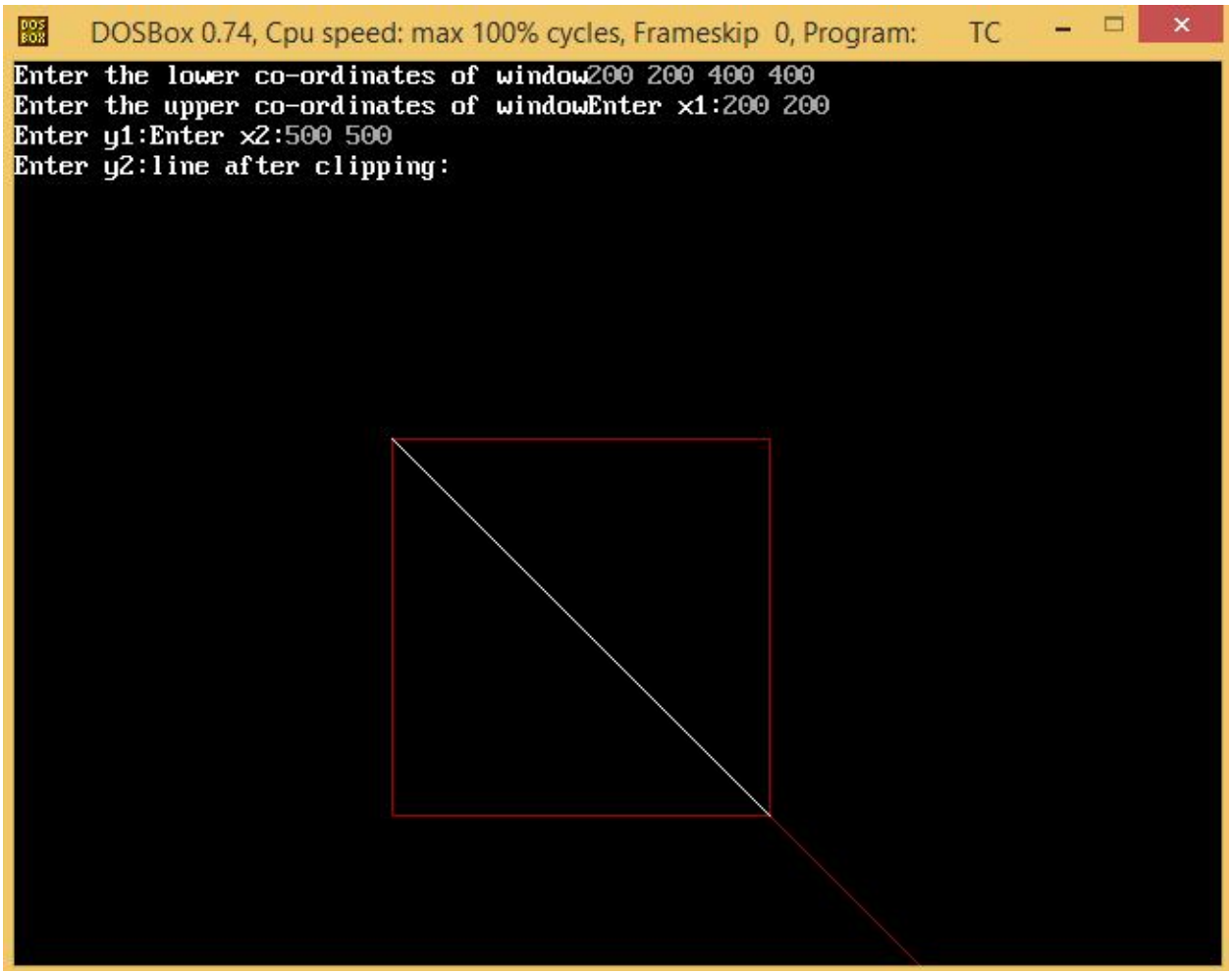
```
                        int temp;
                        //Decide if point1 is inside, if not, calculate intersection
                        if(outcode1==0)
                                temp = outcode2;
                        else
                                temp = outcode1;
                        //Line clips top edge
                        if(temp & TOP){
                                x = x1+ (yh-y1)/m;
                                y = yh;
                        }
                        else if(temp & BOTTOM){          //Line clips bottom edge
                                x = x1+ (yl-y1)/m;
                                y = yl;
                        }else if(temp & LEFT){    //Line clips left edge
                                x = xl;
                                y = y1+ m*(xl-x1);
                        }else if(temp & RIGHT){  //Line clips right edge
                                x = xh;
                                y = y1+ m*(xh-x1);
                        }
                        //Check which point we had selected earlier as temp, and replace its co-
ordinates
                        if(temp == outcode1){
                                x1 = x;
                                y1 = y;
                                outcode1 = getcode(x1,y1);
                        }else{
                                x2 = x;
                                y2 = y;
                                outcode2 = getcode(x2,y2);
                        }
                }
        }
        setcolor(WHITE);
        cout<<"After clipping:";
        if(accept)
                line(x1,y1,x2,y2);
        getch();
        closegraph();
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter bottom left and top right co-ordinates of window: 100 100 200 200
Enter the endpoints of the line: 100 100 300 300
After clipping:

Practical 8b
Aim: Program to implement Liang Barsky Line Clipping Algorithm in C++

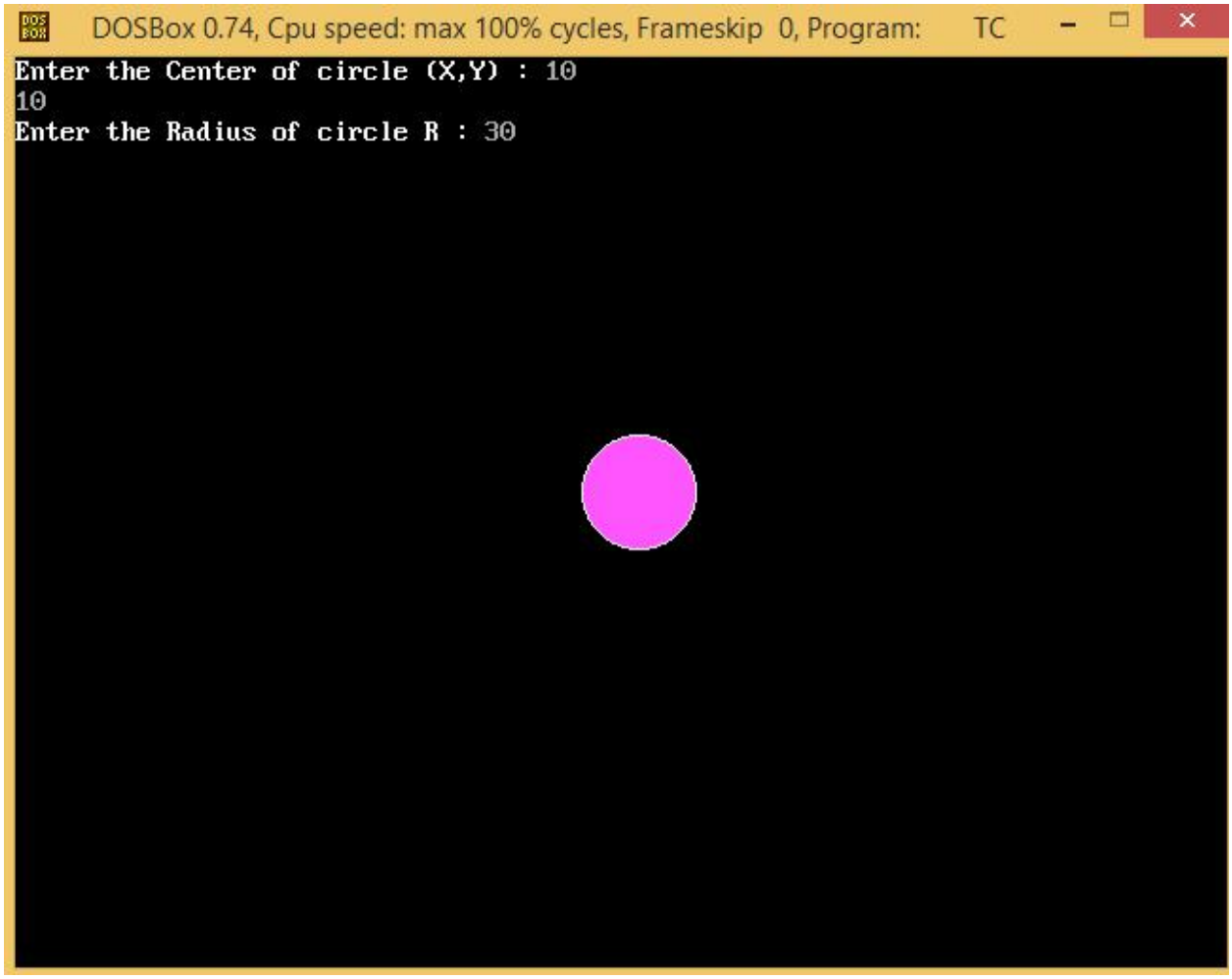
```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
	int gd=DETECT,gm=100;
	initgraph(&gd,&gm,"C:\\TC\\BGI");
	int x1,y1,x2,y2,xmax,xmin,ymax,ymin,xx1,yy1,xx2,yy2,dx,dy,i;
	int p[4],q[4];
	float t1,t2,t[4];
	cout<<"Enter the lower co-ordinates of window";
	cin>>xmin>>ymin;
	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=t[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();
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter the lower co-ordinates of window200 200 400 400
Enter the upper co-ordinates of windowEnter x1:200 200
Enter y1:Enter x2:500 500
Enter y2:line after clipping:

Practical 9a
Aim: Write a program to fill a circle using flood fill algorithm

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

void floodFill(int, int, int, int);
int midx=319, midy=239;

void main()
{
    int gdriver=DETECT, gmode, x,y,r;
    initgraph(&gdriver, &gmode, "c:\\tc\\bgi");

    cleardevice();

    printf("Enter the Center of circle (X,Y) : ");
    scanf("%d %d",&x,&y);
    printf("Enter the Radius of circle R : ");
    scanf("%d",&r);

    circle(midx+x,midy-y,r);
    getch();
    floodFill(midx+x,midy-y,13,0);

    getch();
    closegraph();
}

void floodFill(int x, int y, int fill, int old)
{
    if(getpixel(x,y) == old)
    {
        putpixel(x,y,fill);
        delay(5);
        floodFill(x+1,y,fill,old);
        floodFill(x-1,y,fill,old);
        floodFill(x,y+1,fill,old);
        floodFill(x,y-1,fill,old);
    }
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter the Center of circle (X,Y) : 10
10
Enter the Radius of circle R : 30

Practical 9b
Aim: Program Boundary Fill Algorithm in C++

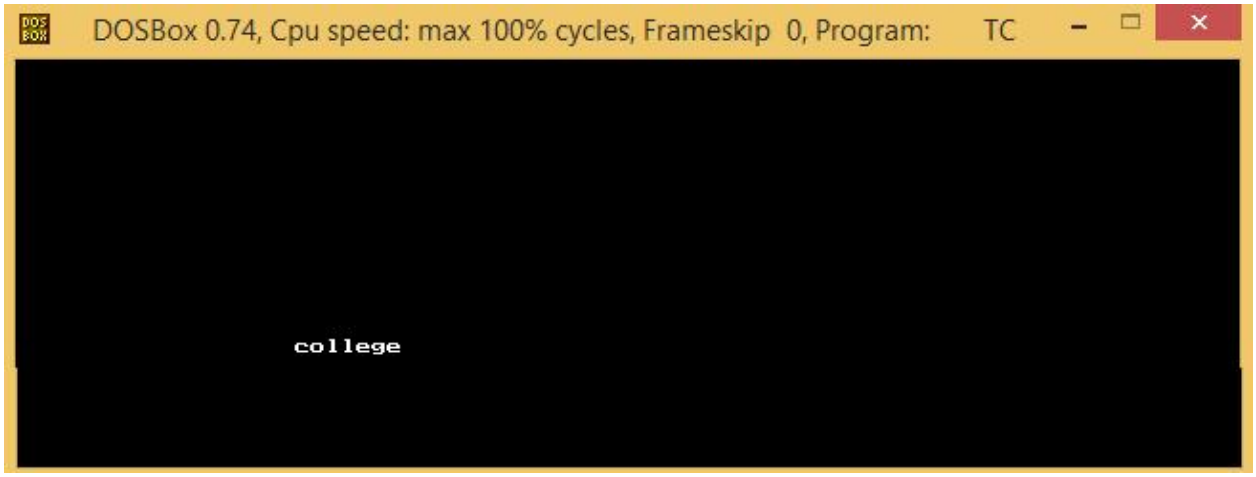
```
#include<graphics.h>
#include<stdlib.h>
#include<iostream.h>
#include<conio.h>
#include<dos.h>
void main()
{
 void boundary_fill(int x,int y,int f,int b);

  int gd = DETECT,gm=100;
  initgraph(&gd, &gm, "C:\\TC\\BGI");

        setcolor(getmaxcolor());

        circle(100, 100, 10);
        boundary_fill(100,100,4,15);

        getch();
        closegraph();
}
void boundary_fill(int x,int y,int f,int b)
{
        if(getpixel(x,y)!=b && getpixel(x,y)!=f)
        {
                putpixel(x,y,f);
                delay(10);
                boundary_fill(x+1,y,f,b);
                boundary_fill(x-1,y,f,b);
                boundary_fill(x,y+1,f,b);
                boundary_fill(x,y-1,f,b);
        }
}
```

Practical 10a

Aim: Text Animation Program Using C++ Programming

```
#include<stdio.h>
#include<math.h>
#include<dos.h>
#include<conio.h>
#include<graphics.h>
#include<iostream.h>
#define round(val) (int)(val+0.5)

void main() {
  int gd = DETECT, gm=100, sx, sy, tx, ty;
initgraph(&gd,&gm,"C:\\TC\\BGI");
  char text[50];

  void move(int, int, int, int, char[]);

  cout<<"Enter the text:";
  cin>>text;
  cout<<"Enter the initial points:";
  cin>>sx>>sy;
  cout<<"Enter the TARGET points:";
  cin>>tx>>ty;

  outtextxy(sx, sy, text);

  move(sx, sy, tx, ty, text);
  getch();
  closegraph();

}

void move(int sx, int sy, int tx, int ty, char text[50]) {

  int dx = tx - sx, dy = ty - sy, steps, k;
  float xin, yin, x = sx, y = sy;

  getch();

  if (abs(dx) > abs(dy))
    steps = abs(dy);
```

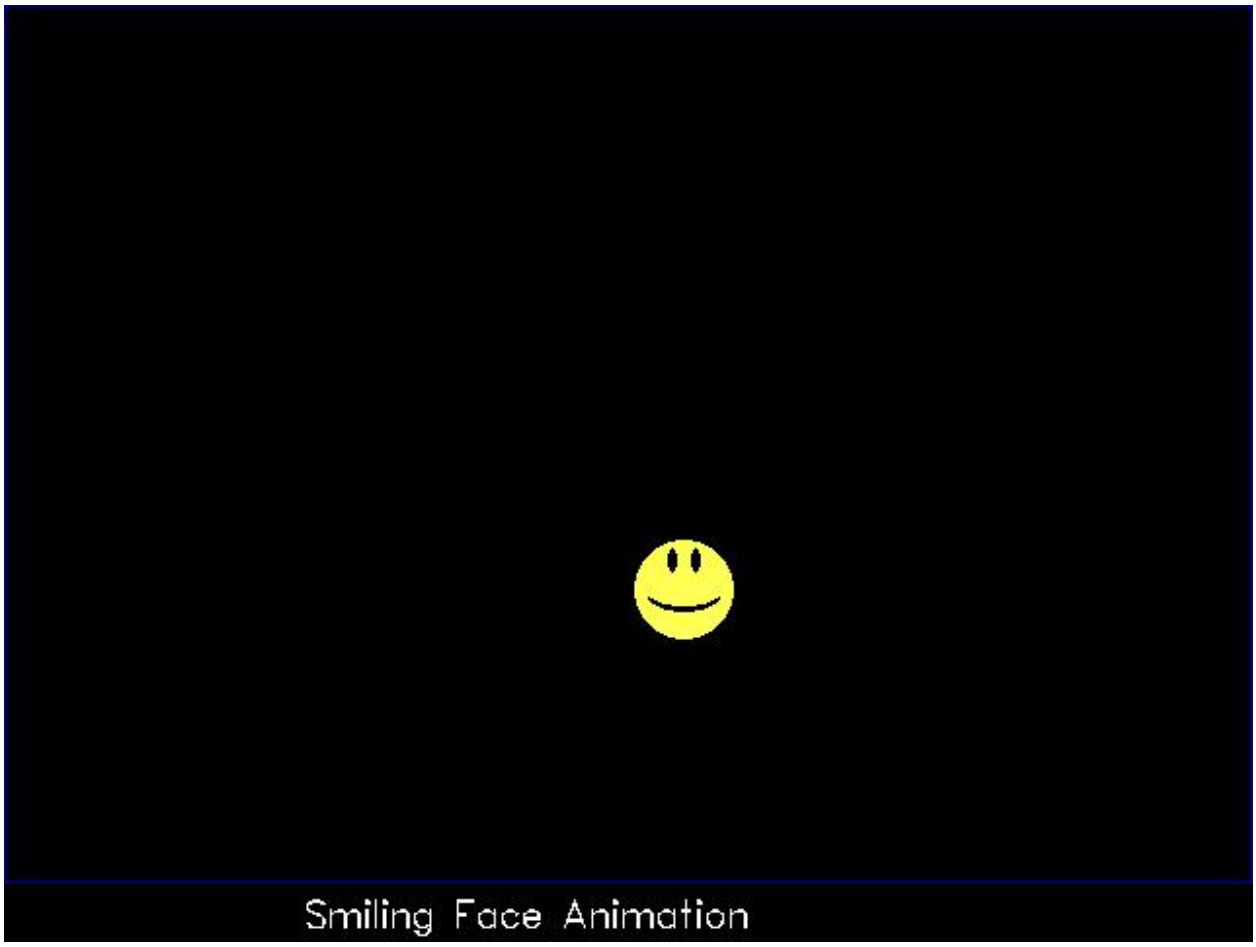
```
  else
    steps = abs(dy);

  xin = dx / (float) steps;
  yin = dy / (float) steps;

  for (k = 0; k < steps; k++) {
    cleardevice();
    x += xin;
    y += yin;
    setcolor(15);
    outtextxy(round(x), round(y), text);
    delay(50);
  }
}
```



Practical 10b
Aim: Smiling face animation program

```
#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();

}
```



Practical 10c

Aim: Practical on Moving a Car in C++

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>'
#include<dos.h>
class Car
{
public:
void design();
};
void Car::design()
{
for(int i=1;i<=448;i++)
{
cleardevice();
arc(100+i,200,0,180,50);
line(120+i,200,120+i,155);
line(121+i,200,121+i,155);
line(80+i,200,80+i,155);
line(79+i,200,79+i,155);
line(50+i,200,150+i,200);
line(10+i,240,35+i,240);
line(190+i,240,165+i,240);
line(65+i,240,135+i,240);
arc(50+i,240,90,180,40);
arc(150+i,240,0,90,40);
circle(50+i,240,5);
circle(50+i,240,15);
circle(150+i,240,5);
circle(150+i,240,15);
delay(5);
}
}
void main()
{
clrscr();
Car obj;

int gd = DETECT,gm=100;
```

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

obj.design();
getch();
}
```