Practical No. 1

C graphics using <graphics.h>

- It can be used to draw different shapes, display text ,change color and many more.
- Using functions of <graphics.h> in TurboC you can make graphics program like animations, projects and games.
- You can draw circles, line and many other geometrical figures.
- The first step in any graphic program is to initialize the graphic driver on the computer using the 'initgraph' method of <graphics.h>.

void initgraph (int *graphicsDriver , int *graphicsMode, char *driverDirectoryPath)

***graphicsDriver**

- It is a pointer to an integer specifying the graphic driver to be used.
- It tells the compiler which graphics driver to use or to automatically detect the driver.
- In all our programs we will use DETECT macro of <graphics.h> library that instruct compiler to autodetect driver.

***graphicsMode**

- It is a pointer to an integer specifying the graphic mode to be used.
- If the *graphicsDriver is set to DETECT then initgraph sets *graphicsMode to highest resolution available for the driver.

- ***driverDirectoryPath**
  - It specifys the directory path where graphicsDriver files are located (BGI).
- **Closegraph()**
  - At the end of our graphics program, we have to unload the graphicsDriver and set the screen back to text by calling closegraph() function.

- **Steps to set library**

  Option -> Linker -> Libraries ->graphicsLibrary

- A program to draw a line

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
   int gd=DETECT,gm;
   initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
   line(100,100,200,200);
   getch();
   closegraph();
}
```

Newly known functions

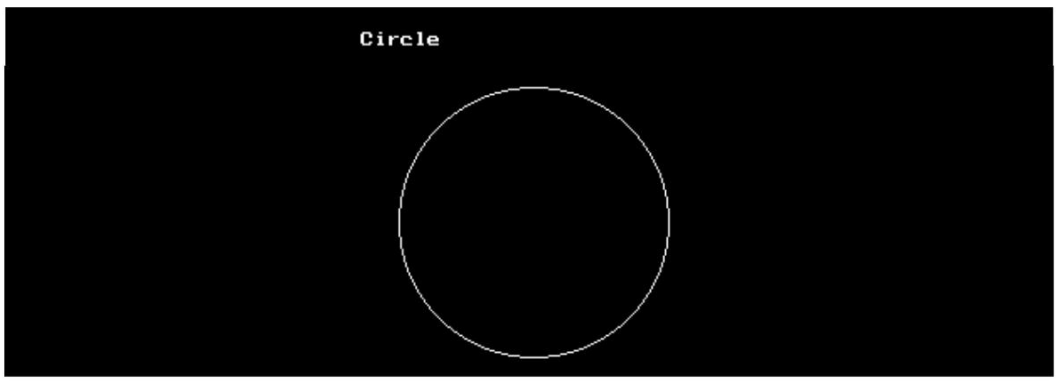
- Line(int x1,int y1,int x2,int y2);

Line() is a library function of <graphics.h> which is used to draw a line from two coordinates .

This function uses 4 parameters x1,y1,x2,y2 that draws a line from point(x1,y1) to point(x2,y2)

- A program to draw a circle at the centre of the screen.

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
	int gd=DETECT,gm;
	int x,y,radius=80;
	initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
	x=getmaxx()/2;
	y=getmaxy()/2;
	outtextxy(x-100,50,"Circle");
	circle(x,y,radius);
	getch();
	closegraph();
}
```



Newly known functions

- getmaxx(void);

  Getmaxx() function returns the maxium X coordinate for current graphics mode and driver.

- getmaxy(void);

  Getmaxy() function returns the maxium Y coordinate for current graphics mode and driver.

- outtextxy (int x, int y, char *string);

  These functions are used to display text on screen in graphics mode. Outtext displays text at the current position while outtexy displays text from specified point(x,y) on the screen.

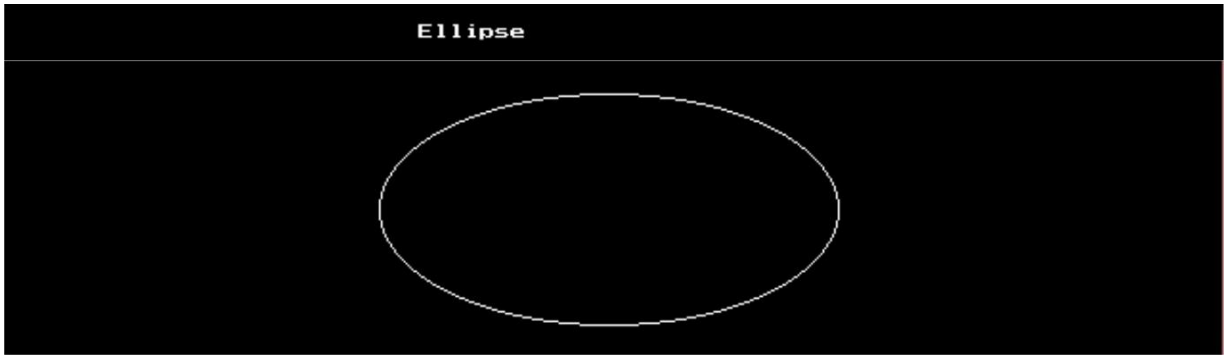
- circle(int x, int y, int radius);

  Circle() draws a circle in the current drawing colour with its center at (x,y) and the radius given by radius.

- A program to draw a ellipse at the centre of the screen.

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
    int gd=DETECT,gm;
    int x,y;
    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
    x=getmaxx()/2;
    y=getmaxy()/2;
    outtextxy(x-100,50,"Ellipse");
    ellipse(x,y,0,360,120,60);
    getch();
    closegraph();
```

}



Newly known functions

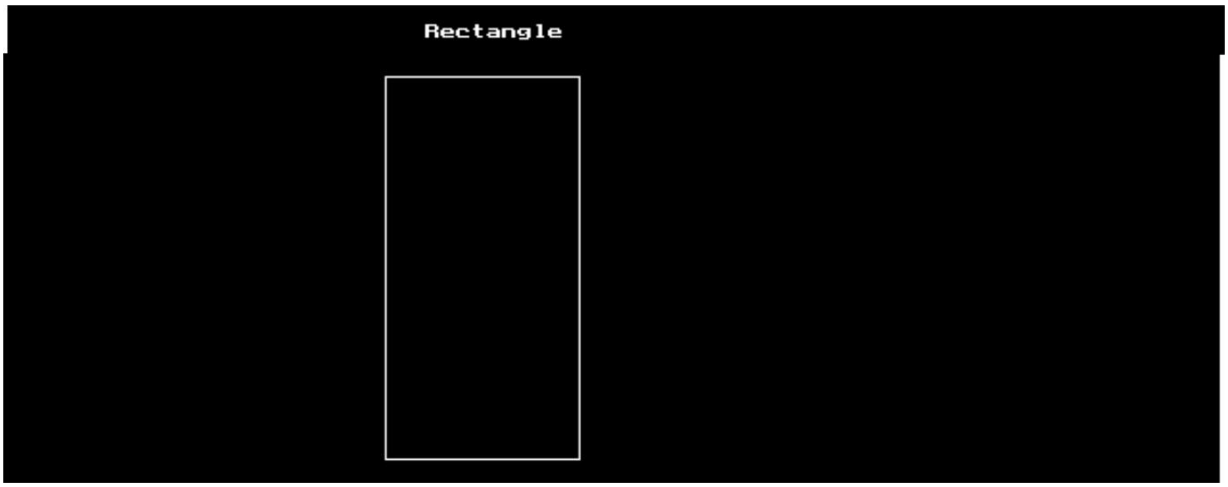
- Ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius):

  It is used to draw an ellipse(x,y) are coordinate of the center of the ellipse, stngle is the starting angle, endangle is the ending angle, and fifth and sixth parameters specifies the X and Y radius.

- A program to draw a rectangle at the centre of the screen.

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
        int gd=DETECT,gm;
    int x,y;
    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
    setcolor(WHITE);
    x=getmaxx()/2;
    y=getmaxy()/2;
    outtextxy(200,50,"Rectangle");
    //rectangle(250,100,350,300);
    rectangle(x-100,y-50,x+100,y+50)
    // rectangle(x-length/2, y-breadth/2, x+ length/2, y+ breadth/2)
```

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



Newly known functions

- rectangle(int left, int top, int right, int bottom);

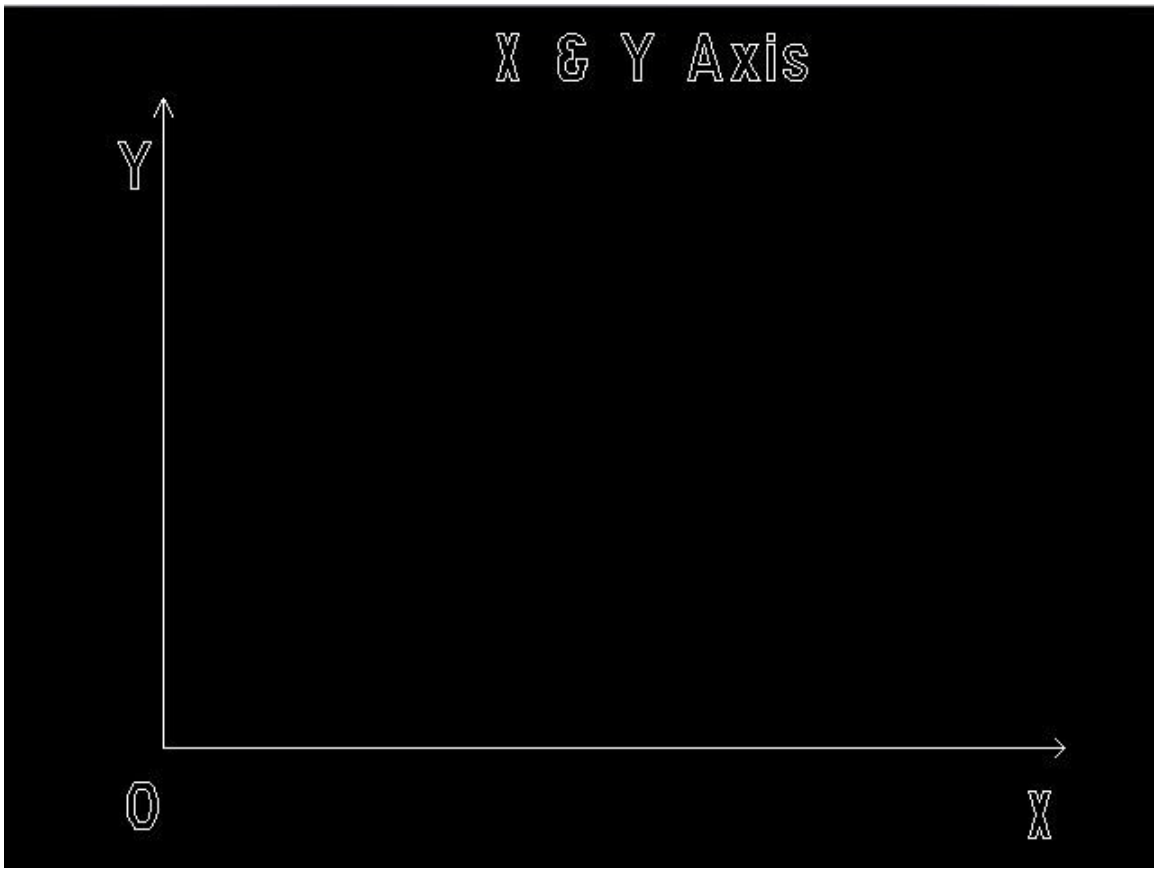
  Rectangle() is used to draw a rectangle. Coordinate of left top and right bottom corner are required to draw the rectangle.

  Left specifies the Y coordinate of top left corner, right specifies the X coordinate of right bottom corner, and bottom specifies the Y coordinate of the right bottom corner.

- A program to draw coordinate axis .

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
   int gd=DETECT, gm, maxx, maxy,
   midx, midy;
   initgraph(&gd,&gm,"C:\\TURBOC3\\BGI"
   );
   settextstyle(BOLD_FONT,HORIZ_DIR,2);
   outtextxy(270,0,”X and Y Axis”);
   setlinestyle(SOLID_LINE,0,2);
```

```
        line(90,410,90,50);
        line(90,410,590,410);
        line(85,60,90,50);
        line(95,60,90,50);
        line(585,405,590,410);
        line(585,412,590,410);
        outtextxy(65,60,”Y”);
        outtextxy(570,420,”X”);
        outtextxy(70,415,”0”);
        getch();
        closegraph();
}
```



Newly known functions

- settextstyle(int font, int direction, int charsize);

Settextstyle() is used to change the way in which the text appears, using it we can modify the size of text, change direction of text and change the font of text.

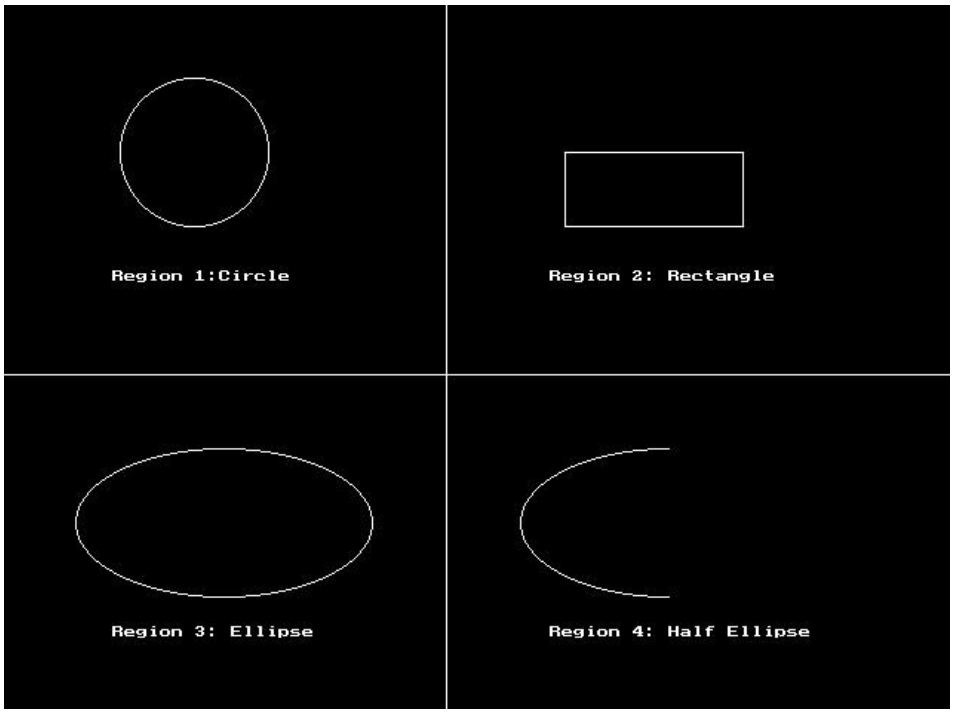
Font argument specifies the font of text, direction can be
HORIZ_DIR (left to right) or direction can be VERT_DIR (bottom to top).

- setlinestyle(int linestyle, unsigned upatter, int thickness);

Setlinestyle() is used to draw the line of different styles. Linestyle argument contains the type of line like solid, dashed or
dotted, etc. upattern parameter is applicable only when type of line is user defined. Thickness specifies the thickness of the line it takes vales 1or 3.

Practical No. 2

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

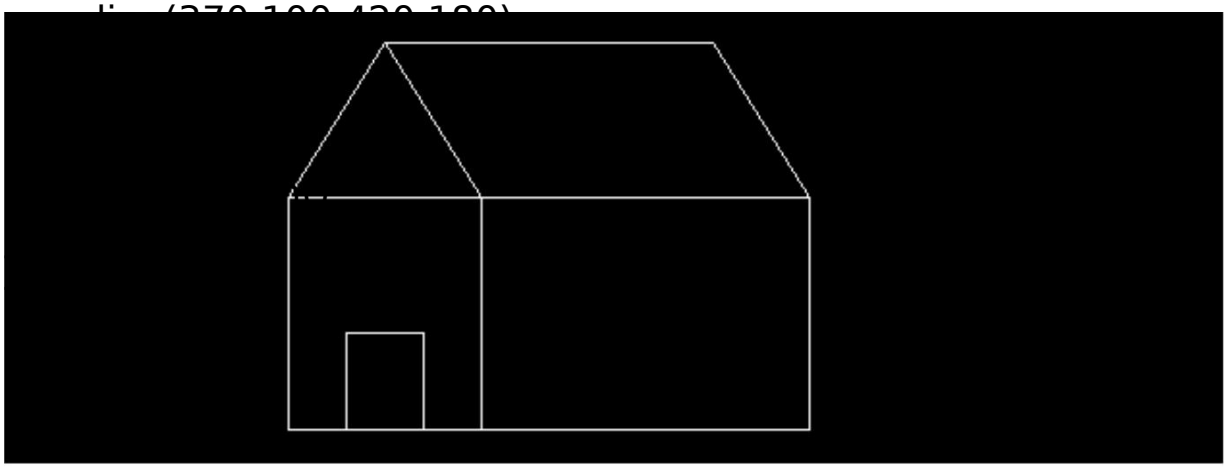


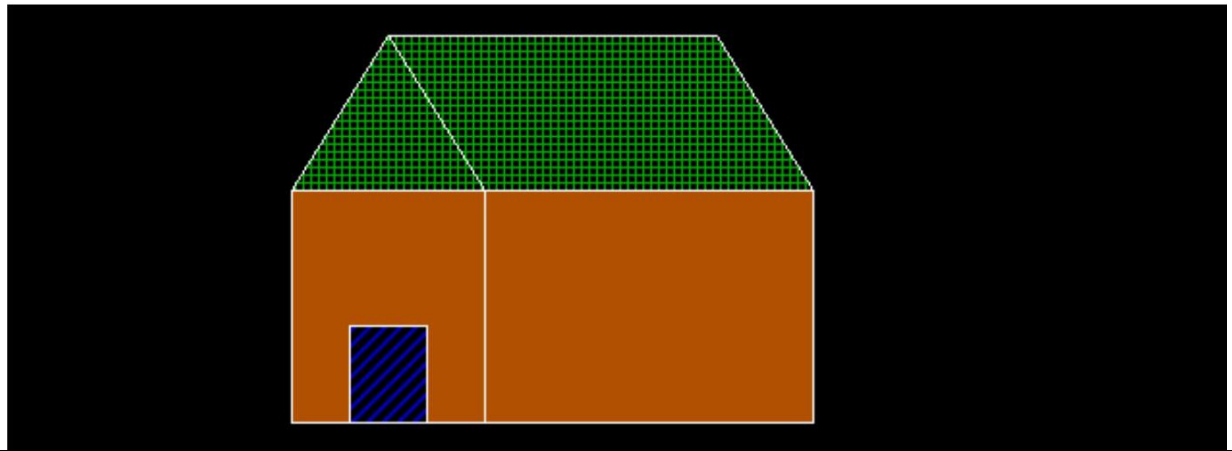

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        int gd=DETECT,gm;
        int x,y,length,breadth ;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        setcolor(WHITE);
        x=getmaxx();
        y=getmaxy();
        line(x/2,0,x/2,y);
        line(0,y/2,x,y/2);
        outtextxy(100,50,"Rectangle!");
        rectangle(50,160,220,90);
        outtextxy(130,260,"Circle");
        circle(150,350,50);
        outtextxy(370,100,"Ellipse!");
        ellipse(500,150,0,360,100,50);
        outtextxy(400,280,"Half Ellpise!");
        ellipse(460,350,0,180,100,50);
```

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

- Draw a simple hut on the screen.

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





Newly known functions

- setcolor(int color);

  Setcolor() is used to set the foreground color in graphics mode.
- setfillstyle(int pattern, int color);

  Setfillstyle() sets the current fill pattern and fill color.
- floodfill(int x, int y, int border);

  Floodfill() is used to fill an enclosed area. Current fill pattern and fill color is used to fill the area. (x,y) is any point on the screen. If (x,y) lies inside the area then inside will be filled otherwise outside will be filled, border specifies the color of boundary of area.

Practical 4A

Newly known functions

- Putpixel(int x, int y, int color);

Putpixel() plots a pixel at a location (x,y) of specified color.

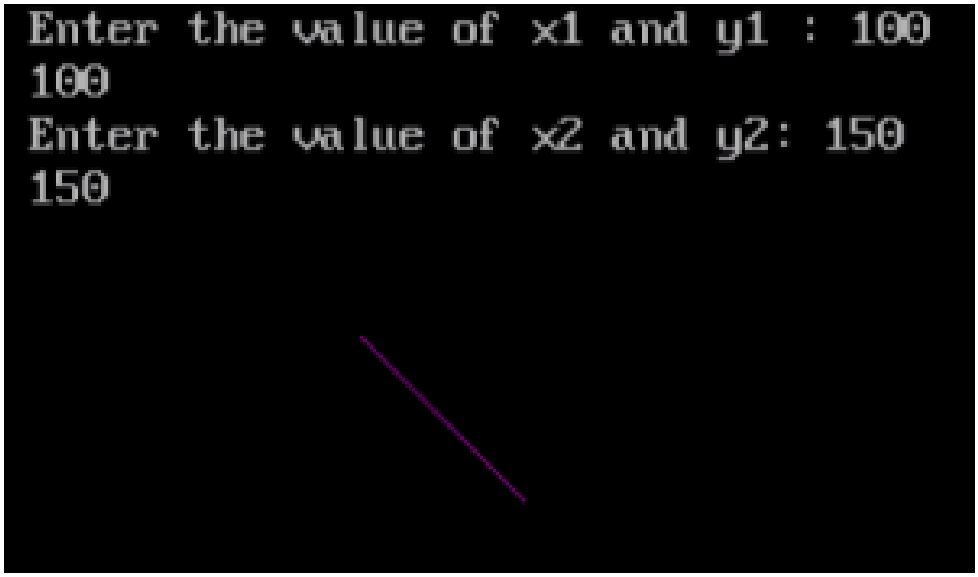
- Delay(unsigned int);

Delay() is used to suspend execution of a program for a particular time. Here, unsigned int is the number of milliseconds.

Program for DDA Line Drawing Algorithm

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

void main( )
{
	float x,y,x1,y1,x2,y2,dx,dy,step;
	int i,gd=DETECT,gm;

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

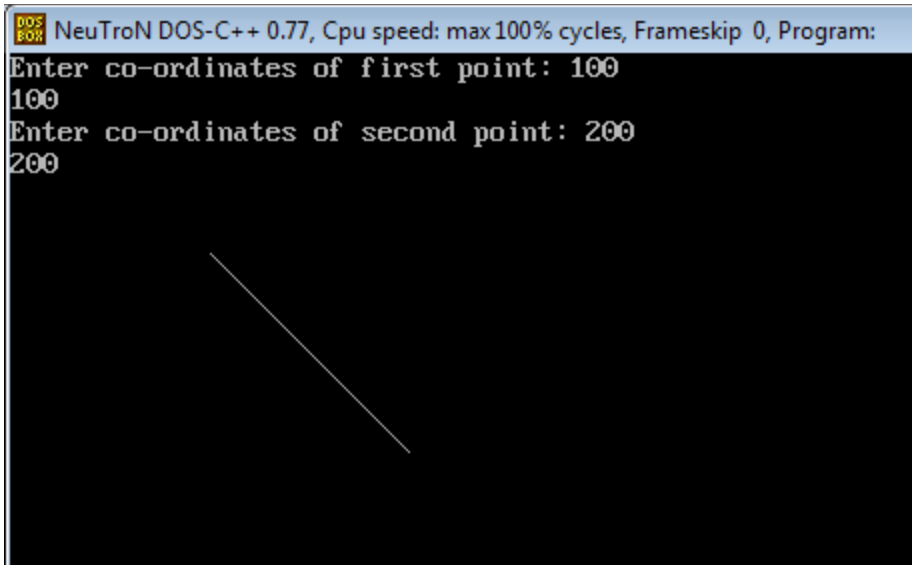


Practical 4B

Program for Bresenham's Line Drawing Algorithm

```
#include<stdio.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;
	}
}

int main()
{
	int gdriver=DETECT, gmode, error, x0, y0, x1, y1;
	initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

	printf("Enter co-ordinates of first point: ");
	scanf("%d%d", &x0, &y0);

	printf("Enter co-ordinates of second point: ");
	scanf("%d%d", &x1, &y1);
	drawline(x0, y0, x1, y1);

	return 0;
}
```

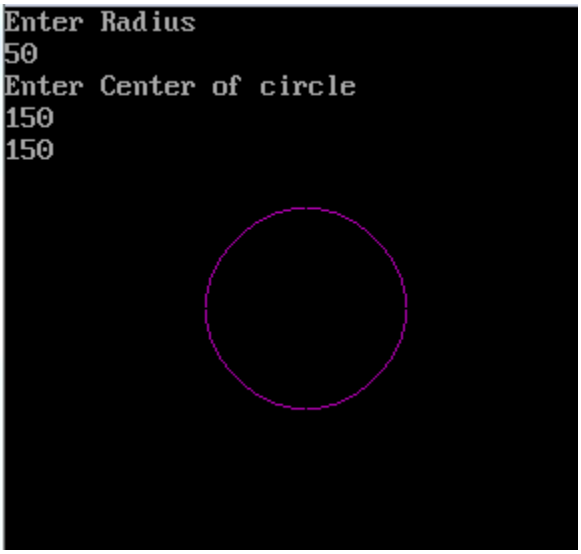


Practical No. 5A

Develop a program for midpoint circle drawing algorithm.

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
int gd=DETECT,gm;
int i,r,x=0,y,xc,yc;
float d;
clrscr();
initgraph(&gd,&gm,"c:\\tc\\");
printf("Enter Radius\n");
scanf("%d",&r);
printf("Enter Center of circle\n");
scanf("%d",&xc);
scanf("%d",&yc);
d=1.25-r;
y=r;
do
{
      if(d<0.0)
      {

            x=x+1;
```

```
            d=d+2*x+1;
      }
      else
      {

            x=x+1;
            y=y-1;
            d=d+2*x-2*y+10;
      }

      putpixel(xc+x,yc+y,5);
      putpixel(xc-y,yc-x,5);      //5 indicate colour(u can write colour
      name also)
      putpixel(xc+y,yc-x,5);
      putpixel(xc-y,yc+x,5);
      putpixel(xc+y,yc+x,5);
      putpixel(xc-x,yc-y,5);
      putpixel(xc+x,yc-y,5);
      putpixel(xc-x,yc+y,5);
      }while(x<y);
getch();
}
```



Practical No. 5B

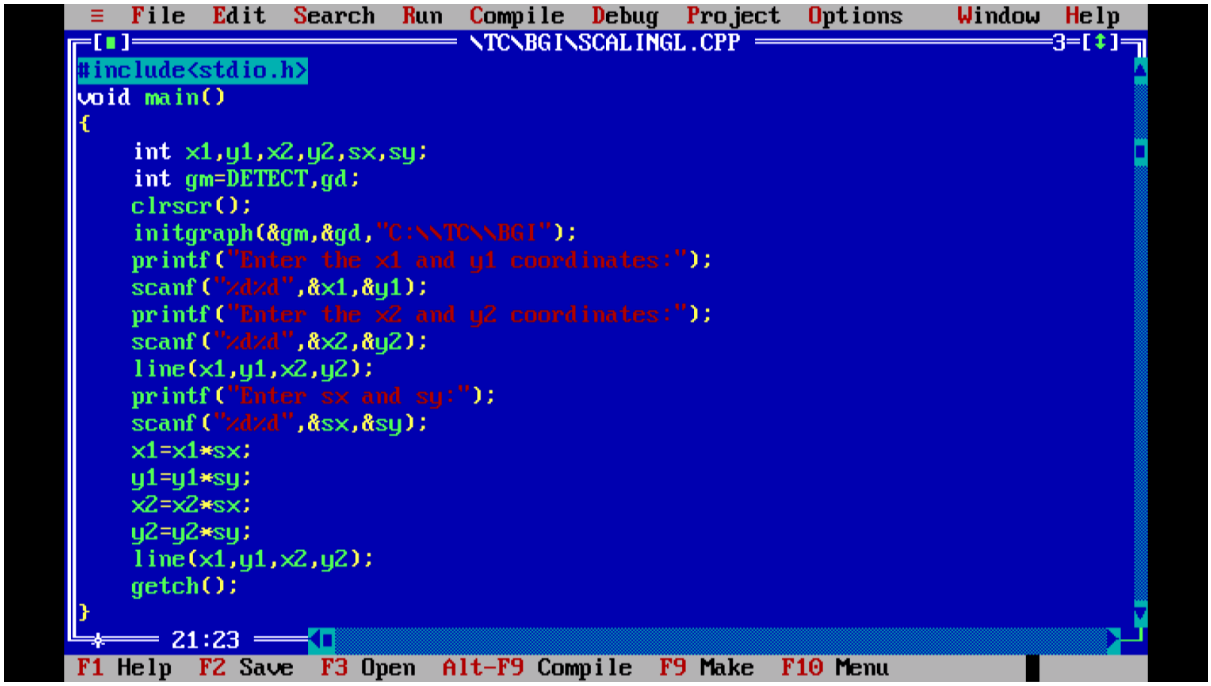
Develop a program for midpoint ellipse drawing algorithm.

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

 void ellipse(int xc,int yc,int rx,int ry)
 {
   int gm=DETECT,gd;
   int x, y, p;
   clrscr();
 initgraph(&gm,&gd,"C:\\TC\\BGI");
   x=0;
   y=ry;
   p=(ry*ry)-(rx*rx*ry)+((rx*rx)/4);
   while((2*x*ry*ry)<(2*y*rx*rx))
   {
       putpixel(xc+x,yc-y,WHITE);
       putpixel(xc-x,yc+y,WHITE);
       putpixel(xc+x,yc+y,WHITE);
       putpixel(xc-x,yc-y,WHITE);

       if(p<0)
       {
           x=x+1;
           p=p+(2*ry*ry*x)+(ry*ry);
       }
       else
       {
           x=x+1;
           y=y-1;
           p=p+(2*ry*ry*x+ry*ry)-(2*rx*rx*y);
       }
   }
   p=((float)x+0.5)*((float)x+0.5)*ry*ry+(y-1)*(y-1)*rx*rx-rx*rx*ry*ry;

        while(y>=0)
   {
       putpixel(xc+x,yc-y,WHITE);
       putpixel(xc-x,yc+y,WHITE);
       putpixel(xc+x,yc+y,WHITE);
```

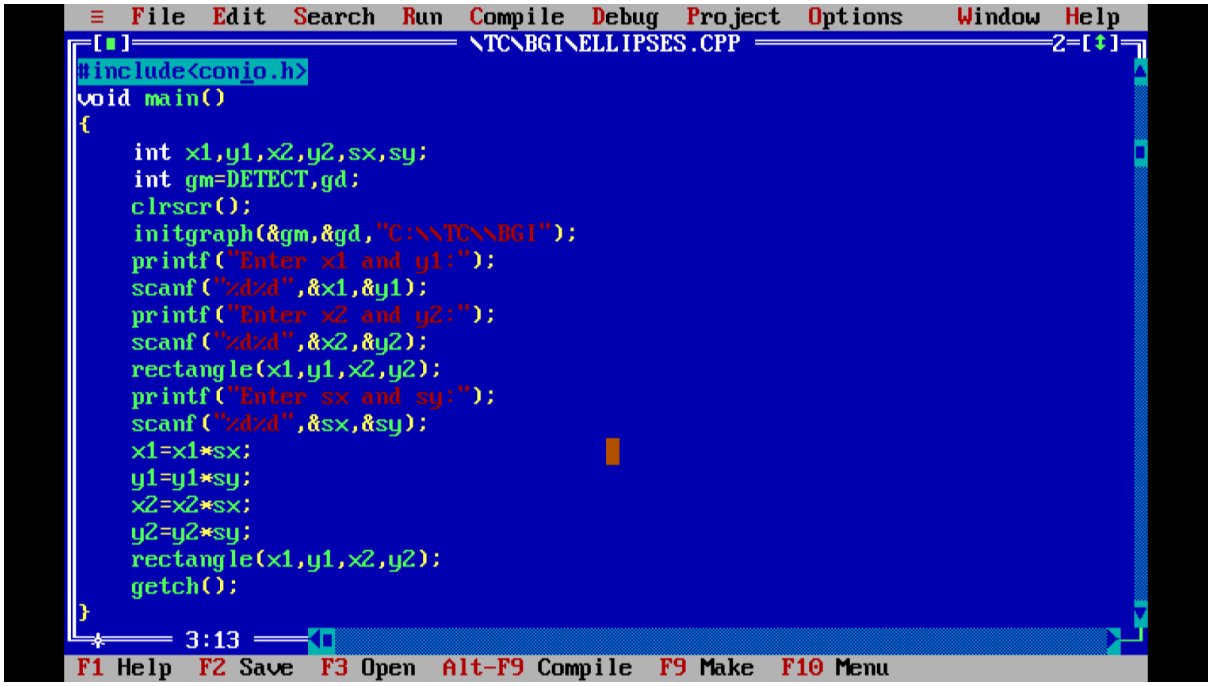
```
        putpixel(xc-x,yc-y,WHITE);

        if(p>0)
        {
            y=y-1;
            p=p-(2*rx*rx*y)+(rx*rx);

        }
        else
        {
            y=y-1;
            x=x+1;
            p=p+(2*ry*ry*x)-(2*rx*rx*y)-(rx*rx);
        }
    }
    getch();
    closegraph();
}

void main()
{
    int xc,yc,rx,ry;
    clrscr();
    printf("Enter Xc=");
    scanf("%d",&xc);
    printf("Enter Yc=");
    scanf("%d",&yc);
    printf("Enter Rx=");
    scanf("%d",&rx);
    printf("Enter Ry=");
    scanf("%d",&ry);
    ellipse(xc,yc,rx,ry);
    getch();
}
```

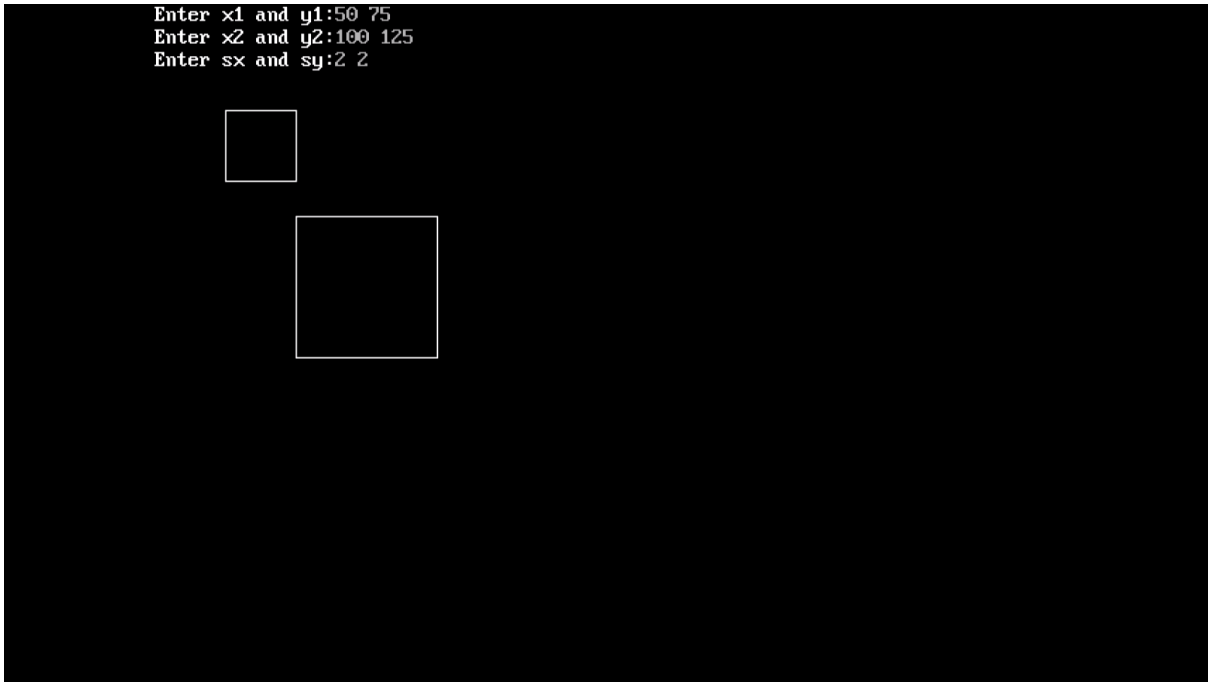
```
Enter Xc=20
Enter Yc=50
Enter Rx=20
Enter Ry=30_
```

Practical 6A:

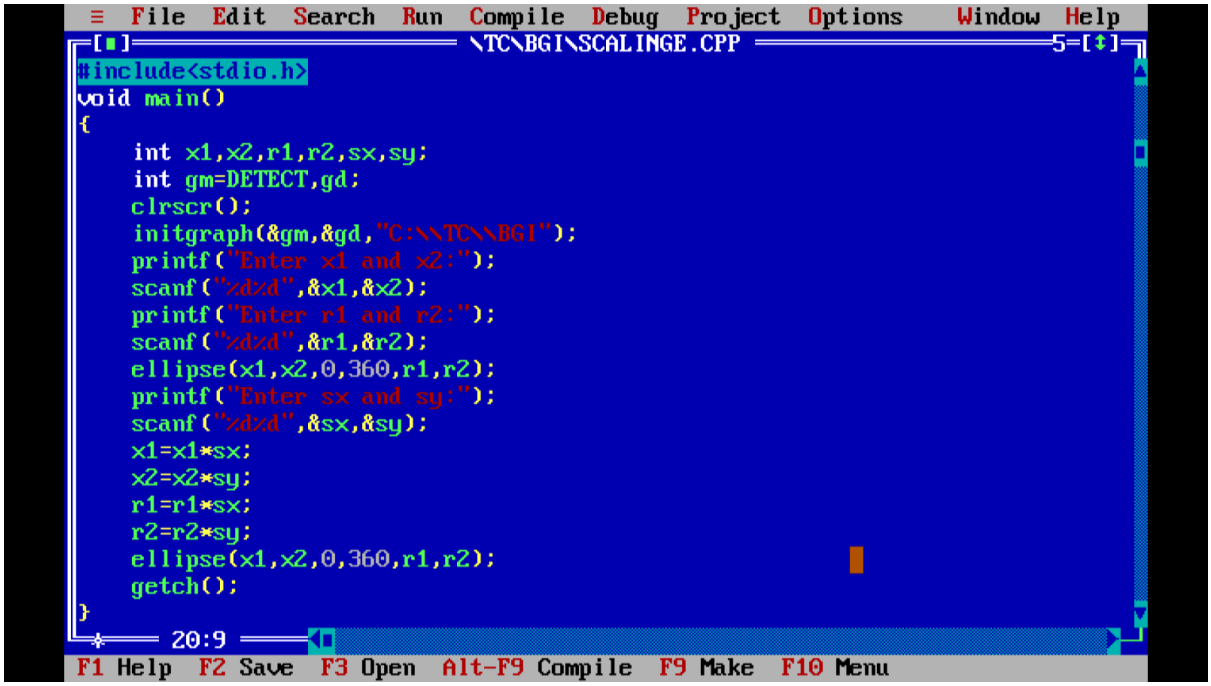


Practical 6B:





Practical 6C:



6A: Scaling line

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        float x1,x2,y1,y2,sx,sy;
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TURBOC4\\TC\\BGI");
        printf("Enter the coordinates x1 and y1:");
        scanf("%f%f",&x1,&y1);
        printf("Enter the coordinates x2 and y2:");
        scanf("%f%f",&x2,&y2);
        line(x1,y1,x2,y2);
        printf("Enter scaling parameters:");
        scanf("%f%f",&sx,&sy);
        printf("Line after scaling:");
        x1=x1*sx;
        y1=y1*sy;
        x2=x2*sx;
        y2=y2*sy;
        line(x1,y1,x2,y2);
        getch();
        closegraph();
}
```

6B: Scaling rectangle

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        float x1,x2,y1,y2,sx,sy;
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TURBOC4\\TC\\BGI");
        printf("Enter the coordinates x1 and y1:");
        scanf("%f%f",&x1,&y1);
        printf("Enter the coordinates x2 and y2:");
        scanf("%f%f",&x2,&y2);
        rectangle(x1,y1,x2,y2);
        printf("Enter scaling parameters:");
        scanf("%f%f",&sx,&sy);
        printf("Square after scaling:");
        x1=x1*sx;
        y1=y1*sy;
        x2=x2*sx;
        y2=y2*sy;
        rectangle(x1,y1,x2,y2);
        getch();
        closegraph();
}
```

6C: Scaling Ellipse

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        float x1,xr,y1,yr,sx,sy;
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TURBOC4\\TC\\BGI");
        printf("Enter the coordinates x1 and y1:");
        scanf("%f%f",&x1,&y1);
        printf("Enter the radius xr and yr:");
        scanf("%f%f",&xr,&yr);
        ellipse(x1,y1,0,360,xr,yr);
        printf("Enter scaling parameters:");
        scanf("%f%f",&sx,&sy);
        printf("Square after scaling:");
        x1=x1*sx;
        y1=y1*sy;
        ellipse(x1,y1,0,360,xr,yr);
        getch();
        closegraph();
}
```

6D: Translation square

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        float x1,x2,y1,y2,sx,sy;
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TURBOC4\\TC\\BGI");
        printf("Enter the coordinates x1 and y1:");
        scanf("%f%f",&x1,&y1);
        printf("Enter the coordinates x2 and y2:");
        scanf("%f%f",&x2,&y2);
        rectangle(x1,y1,x2,y2);
        printf("Enter scaling parameters:");
        scanf("%f%f",&sx,&sy);
        printf("Square after translation:");
        x1=x1+sx;
        y1=y1+sy;
        x2=x2+sx;
        y2=y2+sy;
        rectangle(x1,y1,x2,y2);
        getch();
        closegraph();
}
```

6E: Translation Triangle

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        float x1,y1,x2,y2,x3,y3,tx,ty;
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TURBOC4\\TC\\BGI");
        printf("Enter the coordinates x1 and y1:");
        scanf("%f%f",&x1,&y1);
        printf("Enter the coordinates x2 and y2:");
        scanf("%f%f",&x2,&y2);
        printf("Enter the coordinates x3 and y3:");
        scanf("%f%f",&x3,&y3);
        line(x1,y1,x2,y2);
        line(x1,y1,x3,y3);
        line(x2,y2,x3,y3);
        printf("Enter translation parameters:");
        scanf("%f%f",&tx,&ty);
        printf("Triangle after translation:");
        x1=x1+tx;
        y1=y1+ty;
        x2=x2+tx;
        y2=y2+ty;
        x3=x3+tx;
        y3=y3+ty;
        line(x1,y1,x2,y2);
        line(x1,y1,x3,y3);
        line(x2,y2,x3,y3);
        getch();
        closegraph();
}
```

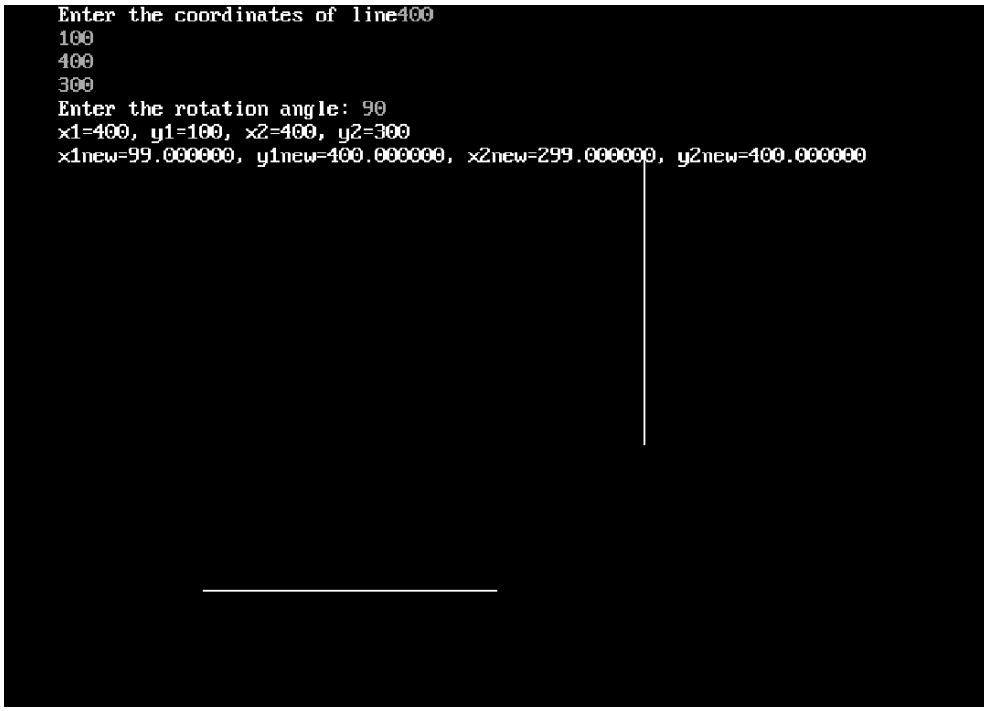
6F: Translation circle

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        float x1,y1,r,tx,ty;
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TURBOC4\\TC\\BGI");
        printf("Enter the coordinates x1 and y1:");
        scanf("%f%f",&x1,&y1);
        printf("Enter radius:");
        scanf("%f",&r);
        printf("enter translation paramteres:");
        scanf("%f%f",&tx,&ty);
        circle(x1,y1,r);
        x1=x1+tx;
        y1=y1+ty;
        circle(x1,y1,r);
        getch();
        closegraph();
}
```

7. Solve the following:

a. Perform 2D Rotation on a given object.

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

#define PI 3.14


void main()
{
	int driver=DETECT, gmode;
	int x1, y1, x2, y2;
	float x1new, y1new, x2new, y2new;
	float r=70;
	float theta;
	fflush(stdin);

	detectgraph(&driver, &gmode);
	initgraph(&driver, &gmode,"c:\\turboc3\\bgi");

	printf("Enter the coordinates of line");
	scanf("%d %d %d %d",&x1,&y1,&x2,&y2);

	printf("Enter the rotation angle: ");
	scanf("%f",&r);
	theta=r*PI/180;
	x1new=x1; y1new=y1;
	x1new=abs(x1*cos(theta)-y1*sin(theta));
	y1new=abs(x1*sin(theta)+y1*cos(theta));

	x2new=abs(x2*cos(theta)-y2*sin(theta));
	y2new=abs(x2*sin(theta)+y2*cos(theta));

	printf("x1=%d, y1=%d, x2=%d, y2=%d",x1,y1,x2,y2);
	printf("\nx1new=%f, y1new=%f, x2new=%f, y2new=%f",x1new,y1new,x2new,y2new);

	line(x1,y1,x2,y2);

	line(x1new,y1new, x2new,y2new);
	getch();

}
```



B. 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.

```
#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,"..\\bgi");
}
void dini()
{
        getch();
        closegraph();
}
void main()
{
```

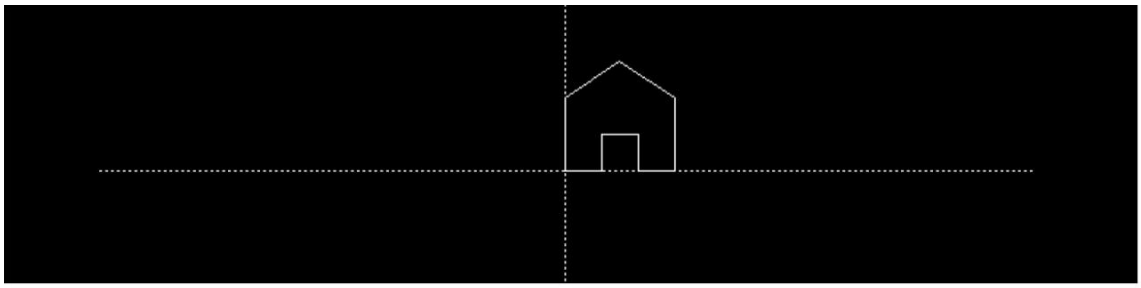
```
int h[9][2],sx,sy,x,y,m,c,choice;
do
{
        clrscr();
        printf("1. Scaling about the origin.\n");
        printf("2. Scaling about an arbitrary point.\n");
        printf("3. Reflection about the line y = mx + c.\n");
        printf("4. Exit\n");
        printf("Enter the choice: ");
  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

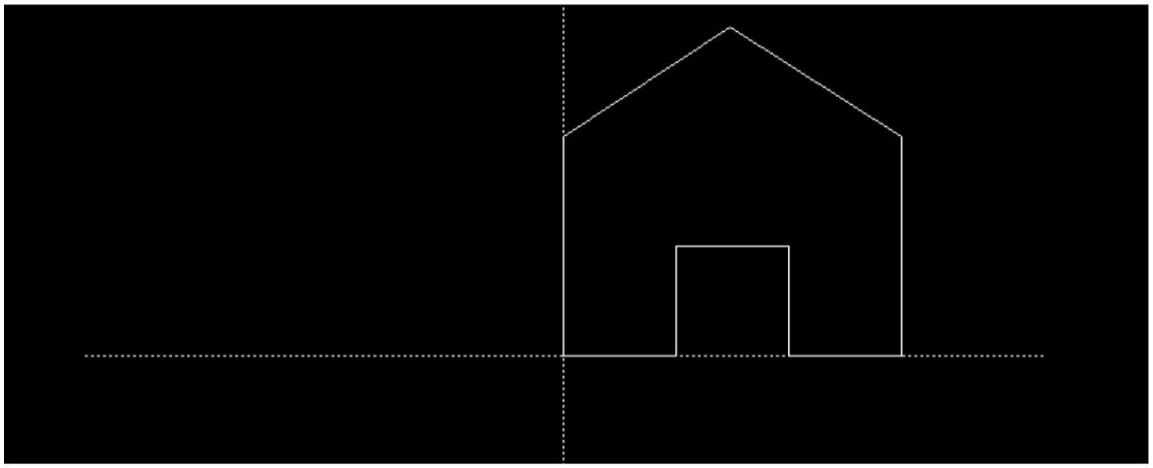
```
        break;

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);
```

```
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 sx and sy: 3 3
```

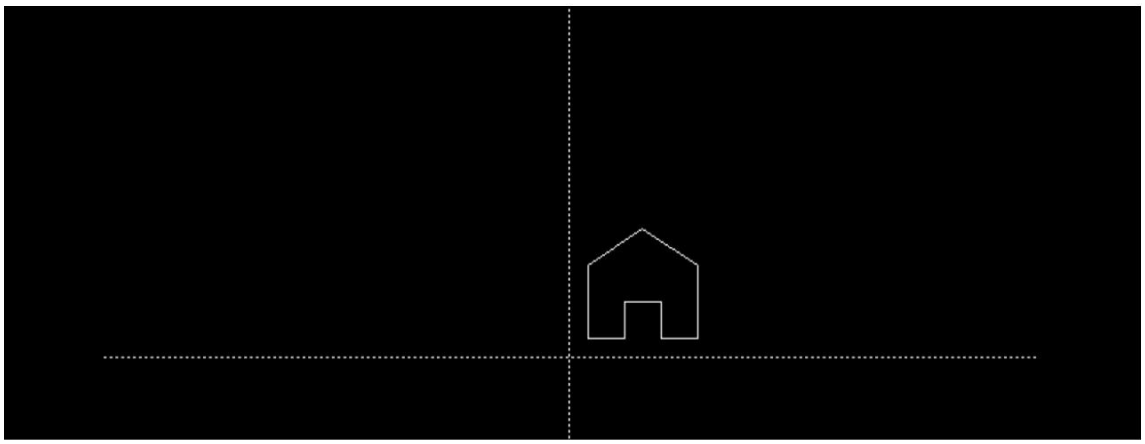
}

```
}while(choi
```



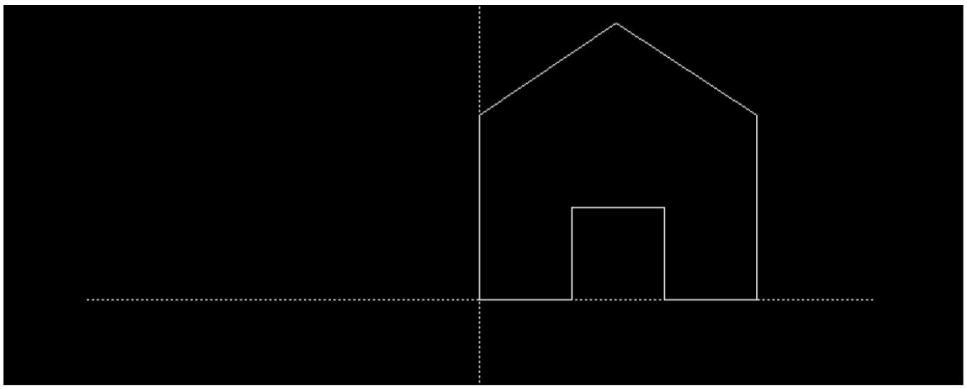


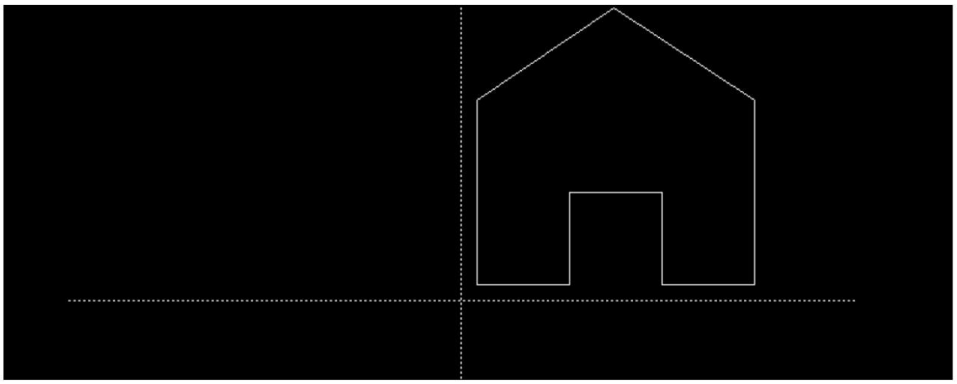
```
1. Scaling about the origin.
2. Scaling about an arbitrary point.
3. Reflection about the line y = mx + c.
4. Exit
Enter the choice: 2
Enter sx and sy: 3 3
Enter the co-ordinates of point(x,y): 13 13_
```



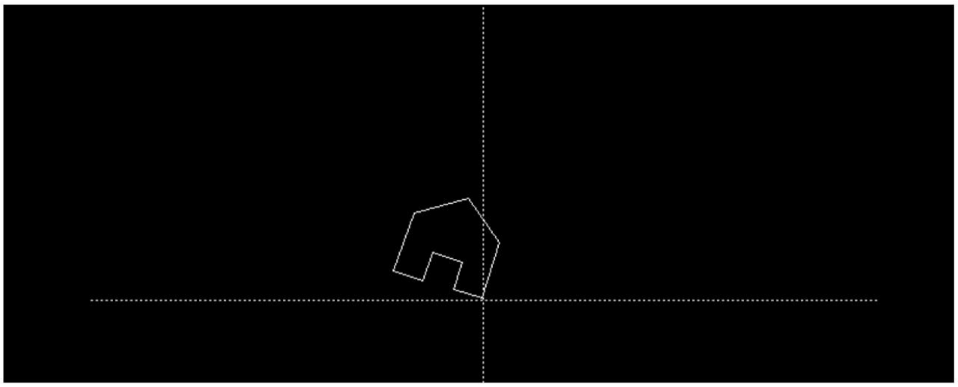
ce!
=4);

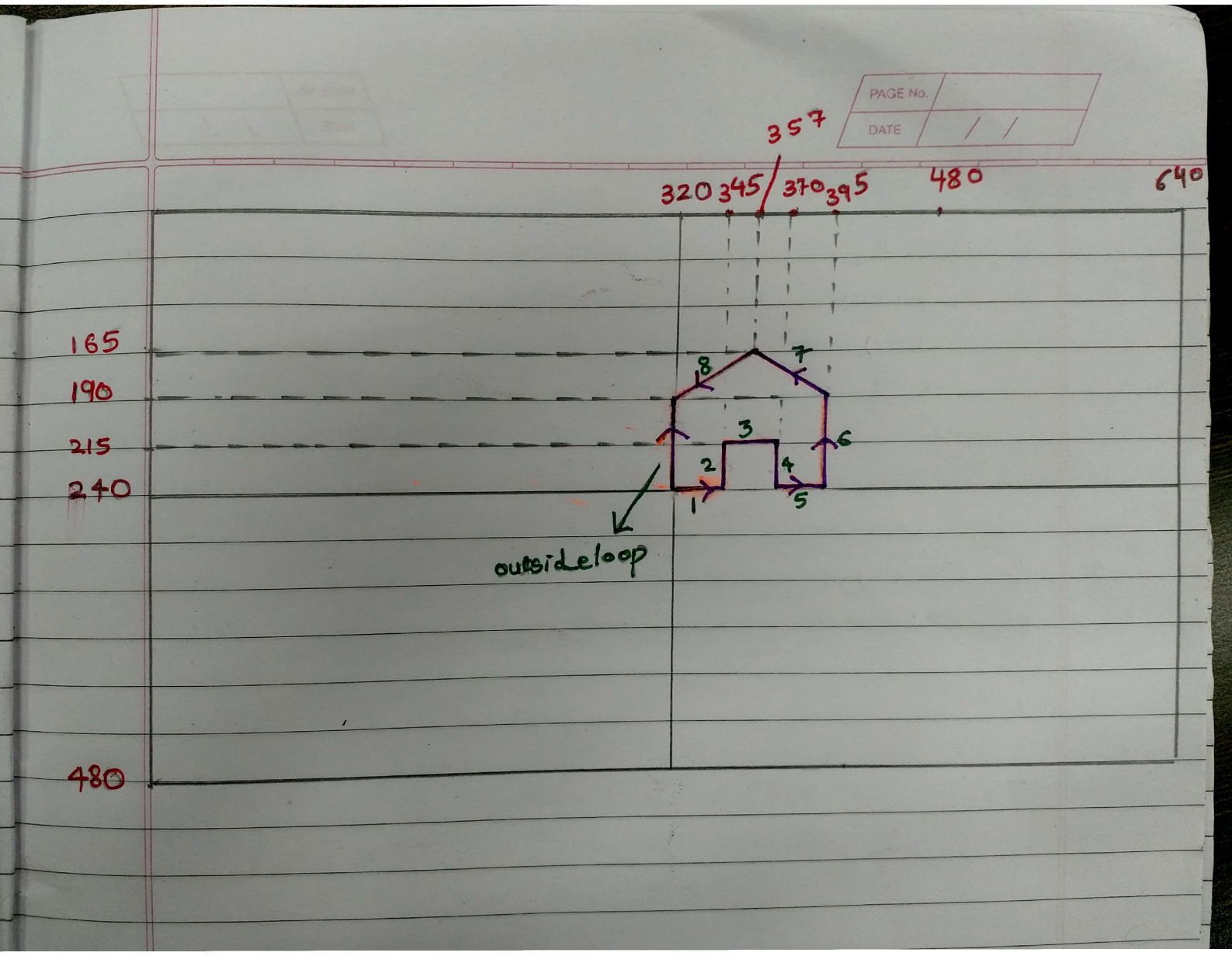
}





```
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
Enter the values of m and c: 6 7
```





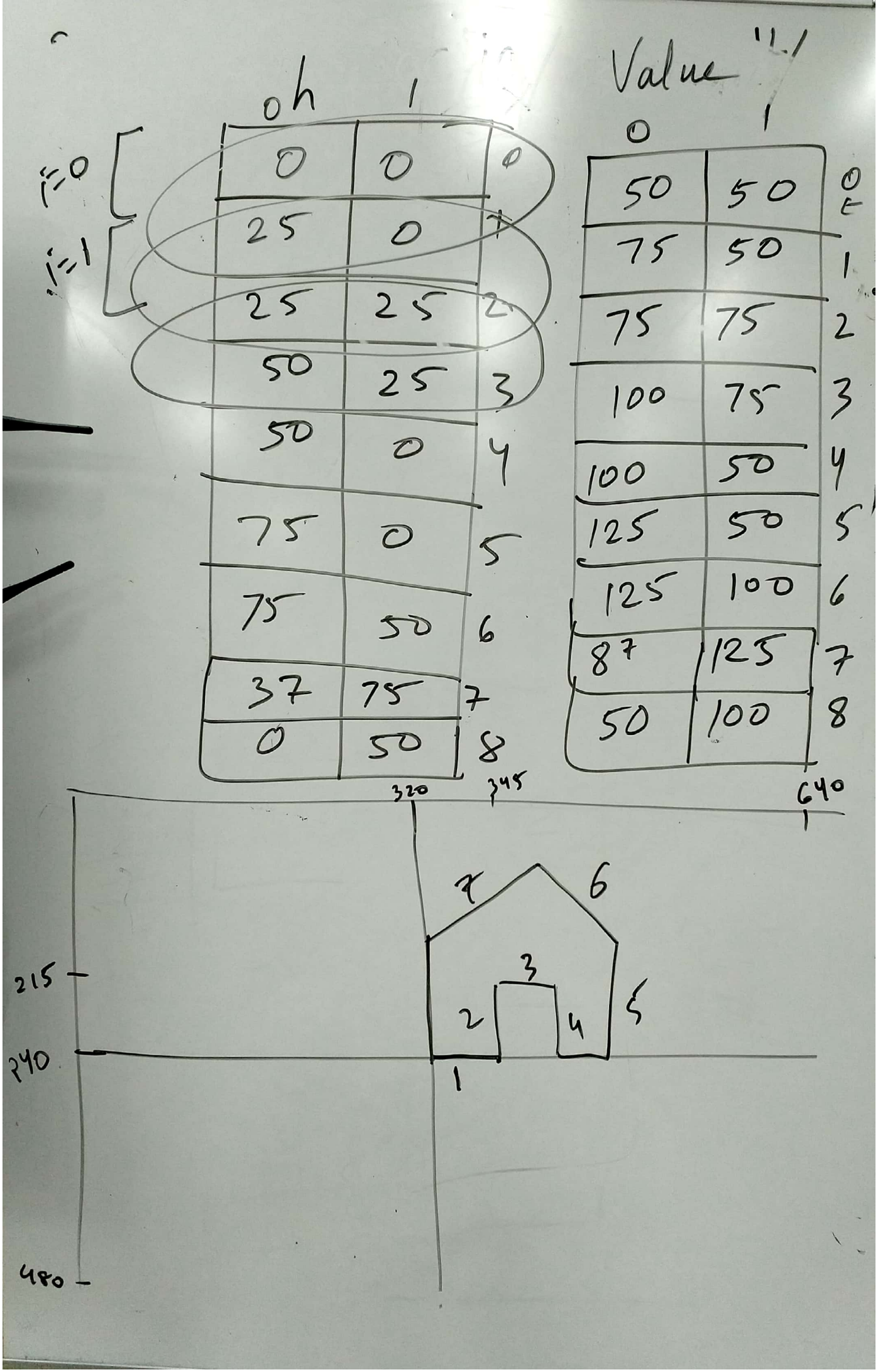
21.02.19

CGA Practical 7-B

		H	0	1	
			0	0	0
i=0	Line (320, 240, 345, 240)	i=0	25	0	1
i=1	Line (345, 240, 345, 215)	i=1	25	25	2
i=2	Line (345, 215, 370, 215)	i=2	50	25	3
i=3	Line (370, 215, 370, 240)	i=3	50	0	4
i=4	Line (370, 240, 395, 240)	i=4	75	0	5
i=5	line (395, 240, 395, 190)	i=5	75	50	6
i=6	Line (395, 190, 357, 165)	i=6	37	75	7
i=7	Line (357, 165, 320, 190)	i=7	0	50	8

Outside for-loop

Line (320, 240, 320, 190)

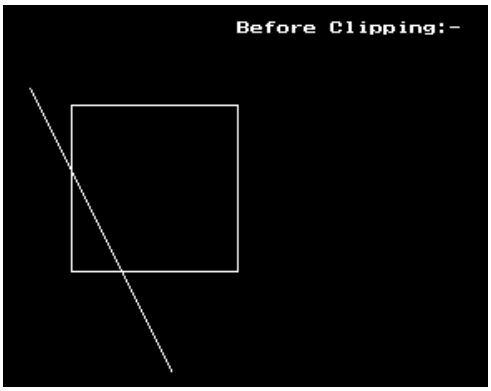


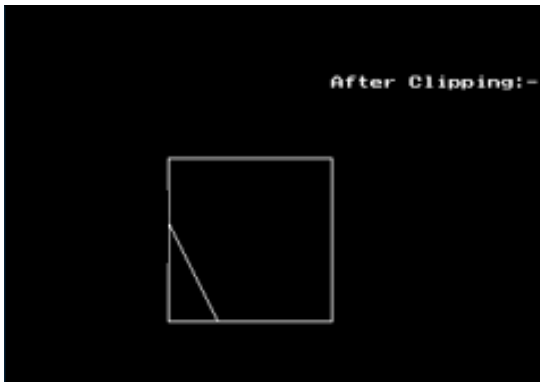
PRACTICAL NO - 8

1. WAP to implement Cohen-Sutherland Clipping :-

```
#include<iostream.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>
int xmin,ymin,xmax,ymax,nx,ny,x1,y1,x2,y2;
void getcode(int x,int y,int code[])
{
  if ( x < xmin)
  code[0] = 1;
  if( x > xmax)
  code[1] = 1;
  if (y < ymin)
  code[2] = 1;
  if (y > ymax)
  code[3] = 1;
}
void getintersects(int code[])
{
float m;
  m = (y2 - y1) / (x2 - x1);
    if(code[0]==1)
    {
      ny = m * (xmin - x1) + y1;
     nx = xmin;
  }
  else if (code[1]==1)
 {
     ny = m * (xmax - x1) + y1;
     nx = xmax;
 }
 if (code[2]==1)
  {
      nx = x1+ (ymin - y1) / m;
      ny = ymin;
  }
else if (code[3]==1)
 {
      nx = x1 + (ymax - y1) / m;
      ny = ymax;
 }

}
void main()
{
```

```
int code1[4]={0,0,0,0}, code2[4]={0,0,0,0},i,draw=0;
int gd=DETECT,gm;
cout<<"C-S Line Clipping Algorithm :";
cout<<"\n Enter window coordinates= ";
cin>>xmin>>ymin>>xmax>>ymax;
cout<<"\n Enter line coordinates=";
cin>>x1>>y1>>x2>>y2;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
outtextxy(200,50,"Before Clipping:-");
rectangle(xmin,ymin,xmax,ymax);
line(x1,y1,x2,y2);
getch();
 cleardevice();
 outtextxy(200,50,"After Clipping:-");
 rectangle(xmin,ymin,xmax,ymax);
  getcode(x1,y1,code1);
  getcode(x2,y2,code2);
for(i=0;i<4;i++)
{
 if(code1[i]==1 && code2[i]==1)
 {
  goto end;
 }
 else
  draw=1;
}
getintersects(code1);
x1=nx;
y1=ny;
getintersects(code2);
x2=nx;
y2=ny;
 if(draw)
  line(x1,y1,x2,y2);
 end:
  getch();
}
```

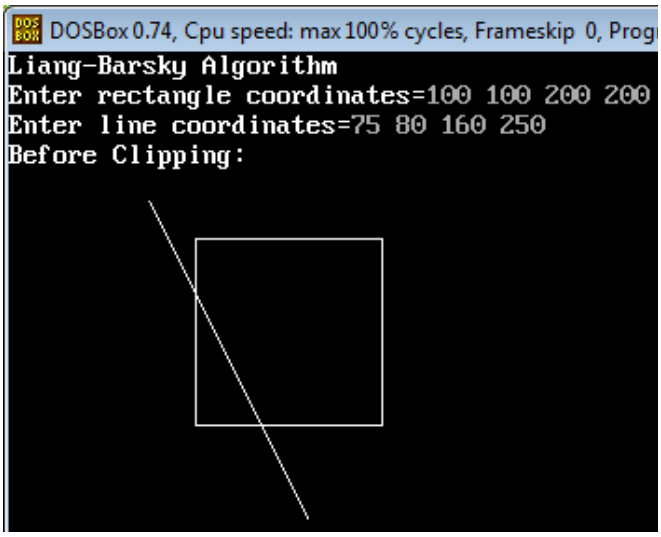


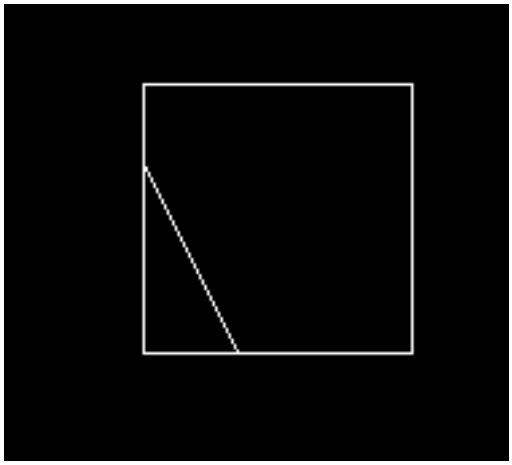


2. WAP to implement Liang-Barsky Line Clipping Algorithm :-

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
int gd,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"c:\\tc\\bgi");
int x1,y1,x2,y2,xmin,xmax,ymin,ymax;
float u1,u2,u3,u4,dx,dy,p1,p2,p3,p4,q1,q2,q3,q4;
cout<<"Liang-Barsky Algorithm";
cout<<"\nEnter rectangle coordinates=";
cin>>xmin>>ymin>>xmax>>ymax;
cout<<"Enter line coordinates=";
cin>>x1>>y1>>x2>>y2;
cout<<"Before Clipping:";
rectangle(xmin,ymin,xmax,ymax);
line(x1,y1,x2,y2);
getch();
cleardevice();
rectangle(xmin,ymin,xmax,ymax);
u1=0.0;
u2=1.0;
dx=x2-x1;
dy=y2-y1;
p1=-dx;
p2=dx;
p3=-dy;
p4=dy;
q1=x1-xmin;
q2=xmax-x1;
q3=y1-ymin;
q4=ymax-y1;
u1=q1/p1;
u2=q2/p2;
u3=q3/p3;
u4=q4/p4;
if(u3>u1 && u3>0)
 u1=u3;
else if(0>u3 && 0>u1)
 u1=0;
if(u4<u2 && u4<1)
 u2=u4;
else if(1<u2 && 1<u4)



u2=1;
if(u1<u2)
{
 x2=x1+(u2*dx);
```

```
 y2=y1+(u2*dy);
 x1=x1+(u1*dx);
 y1=y1+(u1*dy);
 line(x1,y1,x2,y2);
}
getch();
closegraph();
}
```

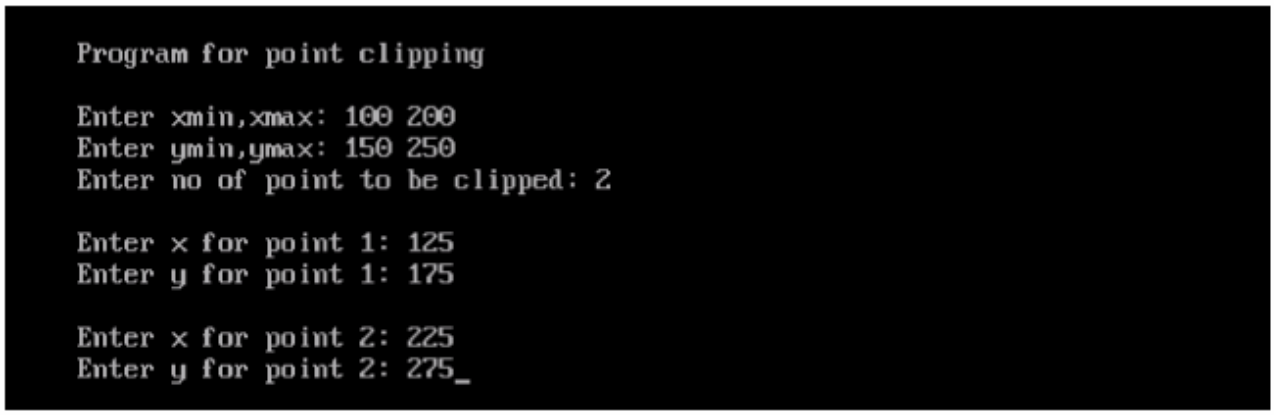




8 a. Write a program to implement point clipping.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void initgraph1()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
}
void main()
{
int xmin,xmax,ymin,ymax;
int x[20],y[20],n,i;
clrscr();
printf("\nProgram for point clipping");
printf("\n\nEnter xmin,xmax: ");
scanf("%d%d",&xmin,&xmax);
printf("Enter ymin,ymax: ");
scanf("%d%d",&ymin,&ymax);
printf("Enter no of point to be clipped: ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter x for point %d: ",i+1);
scanf("%d",&x[i]);
printf("Enter y for point %d: ",i+1);
scanf("%d",&y[i]);
}
initgraph1();
/*Before clipping*/
outtextxy(70,70,"Before Clipping");
rectangle(xmin,ymin,xmax,ymax);
for(i=0;i<n;i++)
{
putpixel(x[i],y[i],14);
}
outtextxy(70,90,"Press key to continue");
getch();
/*After clipping*/
cleardevice();
outtextxy(75,75,"After Clipping");
rectangle(xmin,ymin,xmax,ymax);
for(i=0;i<n;i++)
{
if ((x[i] >= xmin && x[i] <= xmax) && (y[i] >= ymin && y[i] <=
ymax))
{
putpixel(x[i],y[i],14);
}
```

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



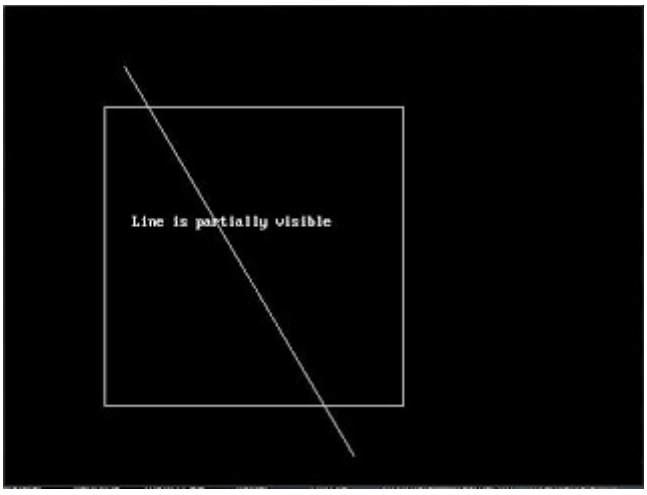
8b Write a program to implement Cohen-Sutherland clipping.

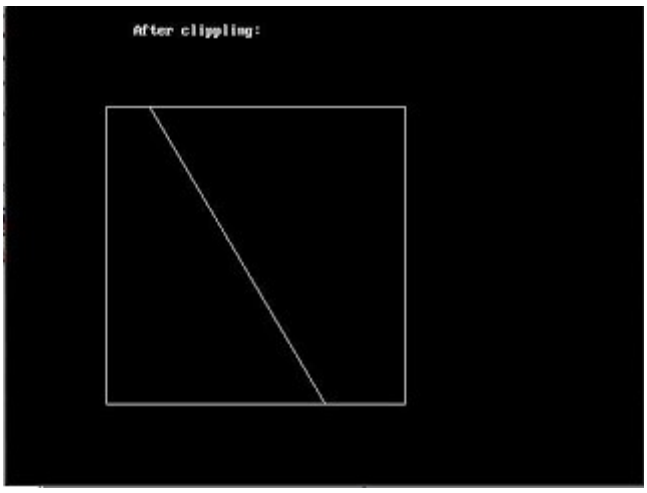
```
#include"stdio.h"
#include"conio.h"
#include"graphics.h"
void main()
{
int gd=DETECT, gm;
float i,xmax,ymax,xmin,ymin,x1,y1,x2,y2,m;
float start[4],end[4],code[4];
clrscr();
initgraph(&gd,&gm,"");
printf("\n\tPlease enter the bottom left co-ordinate of viewport: ");
scanf("%f %f",&xmin,&ymin);
printf("\n\tPlease enter the top right co-ordinate of viewport: ");
scanf("%f %f",&xmax,&ymax);
printf("\nPlease enter the co-ordinates for starting point of line: ");
scanf("%f %f",&x1,&y1);
```

```
printf("\nPlease enter the co-ordinates for ending point of line: ");
scanf("%f %f",&x2,&y2);
for(i=0;i <4;i++)

{
start[i]=0;
end[i]=0;
}
m=(y2-y1)/(x2-x1);
if(x1 <xmin) start[0]=1;
if(x1 >xmax) start[1]=1;
if(y1 >ymax) start[2]=1;
if(y1 <ymin) start[3]=1;
if(x2 <xmin) end[0]=1;
if(x2 >xmax) end[1]=1;
if(y2 >ymax) end[2]=1;
if(y2 <ymin) end[3]=1;
for(i=0;i <4;i++)


code[i]=start[i]&&end[i];

if((code[0]==0)&&(code[1]==0)&&(code[2]==0)&&(code[3]==0))
{
if((start[0]==0)&&(start[1]==0)&&(start[2]==0)&&(start[3]==0)&&(end[0]==0)&
&(end[1]==0)&&(end[2]==0)&&(end[3]==0))
{
cleardevice();
   printf("\n\t\tThe line is totally visible\n\t\tand not a clipping candidate");
   rectangle(xmin,ymin,xmax,ymax);
   line(x1,y1,x2,y2);
   getch();
}
else
{
   cleardevice();
   printf("\n\t\tLine is partially visible");
   rectangle(xmin,ymin,xmax,ymax);
   line(x1,y1,x2,y2);
   getch();

if((start[2]==0)&&(start[3]==1))
{
    x1=x1+(ymin-y1)/m;
    y1=ymin;
}
if((end[2]==0)&&(end[3]==1))
{
    x2=x2+(ymin-y2)/m;
    y2=ymin;
```

```
}
if((start[2]==1)&&(start[3]==0))
{
    x1=x1+(ymax-y1)/m;
    y1=ymax;
}
if((end[2]==1)&&(end[3]==0))
{
    x2=x2+(ymax-y2)/m;
    y2=ymax;
}
if((start[1]==0)&&(start[0]==1))
{
    y1=y1+m*(xmin-x1);
    x1=xmin;
}
if((end[1]==0)&&(end[0]==1))
{
    y2=y2+m*(xmin-x2);
    x2=xmin;
}
if((start[1]==1)&&(start[0]==0))
{
    y1=y1+m*(xmax-x1);
    x1=xmax;
}
if((end[1]==1)&&(end[0]==0))
{
    y2=y2+m*(xmax-x2);
    x2=xmax;
}

clrscr();
cleardevice();
printf("\n\t\tAfter clippling:");
rectangle(xmin,ymin,xmax,ymax);
line(x1,y1,x2,y2);
getch();
}
}
else
{
   clrscr();
   cleardevice();
   printf("\nLine is invisible");
   rectangle(xmin,ymin,xmax,ymax);
}
getch();
closegraph();
}
```

Input:
Please enter the bottom left co-ordinate of viewport: 100 100
Please enter the top right co-ordinate of viewport: 400 400
Please enter the coordinates for starting point of line: 120 60
Please enter the coordinates for ending point of line: 350 450
Output:



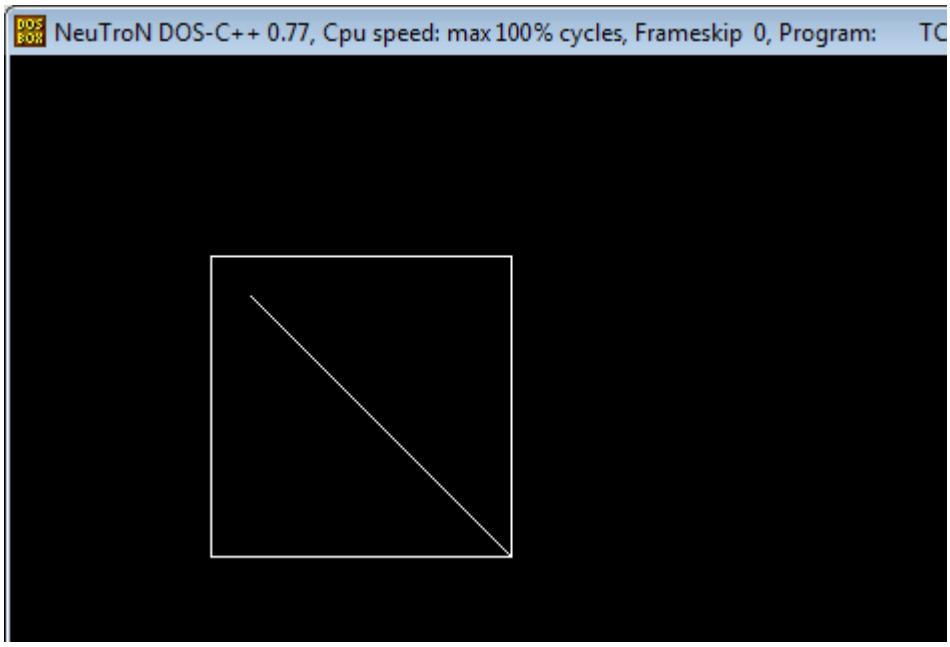


8c. Write a program to implement Liang - Barsky Line Clipping Algorithm

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

x1=120;
y1=120;
x2=300;
y2=300;

xmin=100;
ymin=100;
xmax=250;
ymax=250;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");
rectangle(xmin,ymin,xmax,ymax);
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)
        {
                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 + t2 * p[3];
        line(xx1,yy1,xx2,yy2);
}

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


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program:
TC

Practical No. 9

Flood and Boundary fill algorithm.

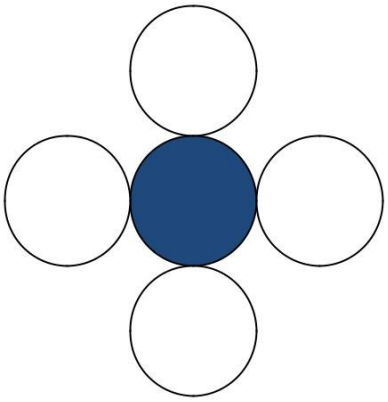
Flood Fill Algorithm

Sometimes we want to fill in (recolor) an area that is not defined within a single color boundary. We paint such areas by replacing a specified interior color instead of searching for a boundary color value. This approach is called a flood-fill algorithm.

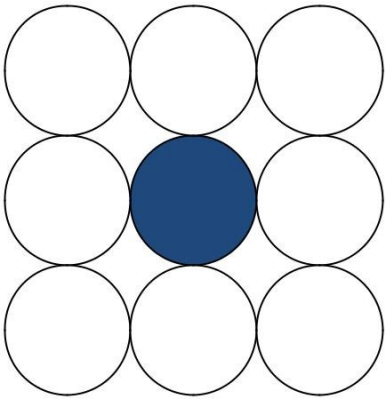
1. We start from a specified interior pixel (x, y) and reassign all pixel values that are currently set to a given interior color with the desired fill color.

2.If the area has more than one interior color, we can first reassign pixel values so that all interior pixels have the same color.

3.Using either 4-connected or 8-connected approach, we then step through pixel positions until all interior pixels have been repainted.



4- connected



8-connected

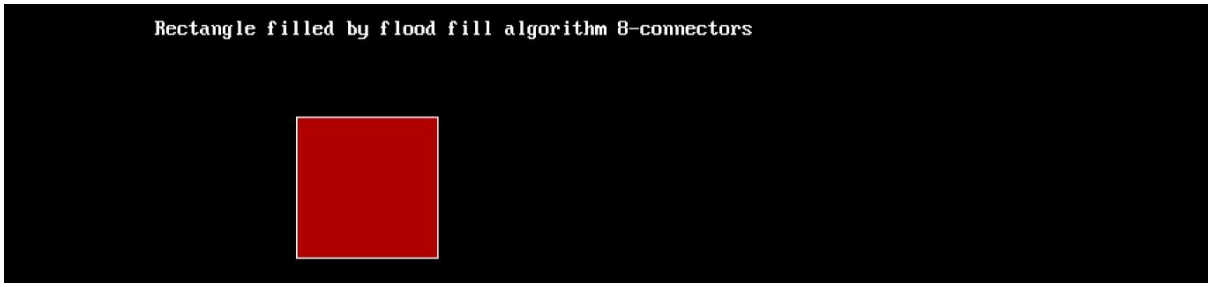
i. Write a program to fill a circle using flood fill algorithm. a. 4-connector

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void flood(int,int,int,int);
void main()
{
   int gd=DETECT, gm;
   clrscr();
   initgraph(&gd,&gm,"C:\\Turboc3\\bgi");
   printf("\n\n\n\n\n\n\nCircle filled by flood fill algoirthm 4-
   connected
");
   circle(100,200,40);
   flood(104,204,8,0);
   getch();
}
void flood (int x, int y, int fill_col, int old_col)
{
   if((getpixel(x,y)==old_col))
   {
         delay(1);
         putpixel(x,y,fill_col);
         flood(x+1,y,fill_col,old_col);
         flood(x-1,y,fill_col,old_col);
         flood(x,y+1,fill_col,old_col);
         flood(x,y-1,fill_col,old_col);
   }
}
```

b. 8-connecter

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void flood(int,int,int,int);
void main()
{
	int gd=DETECT, gm;
	clrscr();
	detectgraph(&gd,&gm);
	initgraph(&gd,&gm,"C:\\Turboc3\\bgi");
	printf("Rectangle filled by flood fill : 8 - connector ");
	rectangle(50,100,150,200);
	flood(54,104,8,0);
	getch();
}
void flood (int x, int y, int fill_col, int old_col)
{
	if((getpixel(x,y)==old_col))
	{
		delay(1);
		putpixel(x,y,fill_col);
		flood(x+1,y,fill_col,old_col);
		flood(x-1,y,fill_col,old_col);
		flood(x,y+1,fill_col,old_col);
```

```
            flood(x,y-1,fill_col,old_col);
            flood(x+1,y-1,fill_col,old_col);
            flood(x+1,y+1,fill_col,old_col);
            flood(x-1,y-1,fill_col,old_col);
            flood(x-1,y+1,fill_col,old_col);
      }
}
```



Boundary Fill Algorithm

The following steps illustrate the idea of recursive boundary fill algorithm

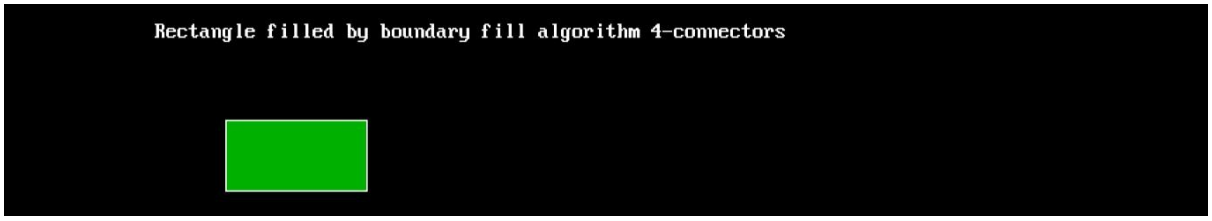
1. Start from an interior point.
2. If the current pixel is not already filled and if it is not an edge point then, set the pixel with fill color and store its neighboring pixels (4 or 8 connected). Store only those neighboring pixels that are not already filled and are not an edge point.
3. Select the next pixel from the stack and continue with step 2.

ii. Write a program to fill a circle using boundary fill algorithm.

a. 4- connector

```
#include<conio.h>
#include<conio.h>
#include<graphics.h>
void boundary_fill(int x, int y, int fcolor, int bcolor)
{
      if ((getpixel(x,y)!=bcolor) && (getpixel(x,y)!= fcolor))
      {
          / delay(10);
            putpixel(x,y,fcolor);
            boundary_fill(x+1,y,fcolor,
            bcolor);      boundary_fill(x-
            1,y,fcolor,bcolor);
            boundary_fill(x,y+1,fcolor,
            bcolor);    boundary_fill(x,y-
            1,fcolor,bcolor);
```

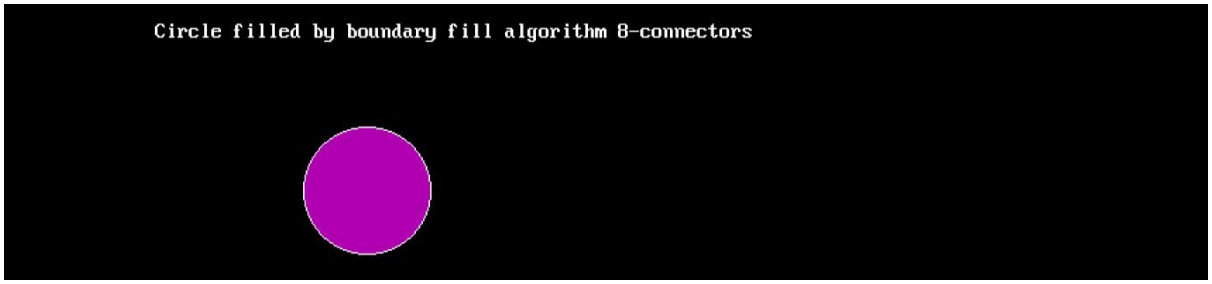
```
        }
}
void main()
{
        int x,y,fcolor,bcolor;
        int gd=DETECT, gm;
        initgraph(&gd,&gm,"C:\\Turboc3\\bgi");
        printf("\nEnter the seed point(x,y): ");
        scanf("%d%d",&x,&y);
        printf("\nEnter boundary color: ");
        scanf("%d",&bcolor);
        printf("\nEnter new color: ");
        scanf("%d",&fcolor);
        circle(100,150,45);
        boundary_fill(x,y,fcolor,bcolor);
        getch();
}
```



b. 8-connector

```
#include<conio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void boundary_fill(int x, int y, int fcolor, int bcolor)
{
	if((getpixel(x,y)!=bcolor)&&(getpixel(x,y)!=fcolor))
	{
		delay(1);
		putpixel(x,y,fcolor);
		boundary_fill(x+1,y,fcolor,bcolor);
		boundary_fill(x-1,y,fcolor,bcolor);
		boundary_fill(x,y+1,fcolor,bcolor);
		boundary_fill(x,y-1,fcolor,bcolor);
		boundary_fill(x+1,y+1,fcolor,bcolor);
		boundary_fill(x+1,y-1,fcolor,bcolor);
		boundary_fill(x-1,y+1,fcolor,bcolor);
		boundary_fill(x-1,y-1,fcolor,bcolor);
	}
}
void main()
{
	int x,y,fcolor,bcolor;
	int gd=DETECT,gm;
	initgraph(&gd,&gm,"C:\\Turboc3\\bgi");
	printf("\nEnter the seed point(x,y): ");
	scanf("%d%d",&x,&y);
	printf("\nEnter boundary color: ");
	scanf("%d",&bcolor);
	printf("\nEnter new color: ");
```

```
        scanf("%d",&fcolor);
        ellipse(100,150,0,360,45,45);
        boundary_fill(x,y,fcolor,bcolor);
        getch();
}
```



Practical No. 10

Animation

i. Write a program to develop a screen saver using graphics function.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
void main ()
{
	int
gd=DETECT,gm,col=480,row=640,font=4,direction=2,size=8,col
or=15;
	initgraph(&gd,&gm,"C:\\TurboC3\\bgi");
	cleardevice();
	while(!kbhit())
	{
		settextstyle(random(font),random(direction),random(siz
		e));
		setcolor(random(color));
		outtextxy(random(col),random(row),"Animation");
		delay(300);
	}
	closegraph();
}
```


Animation
Animation
Animation
Animation
Animation
Animation

ii. Write a program to perform smiling face animation using graphics function.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
void main ()
{
	int gd=DETECT,gm,x=250,y=250;
	initgraph(&gd,&gm,"C:\\TurboC3\\bgi");
	cleardevice();
	while(!kbhit())
	{
		ellipse(x,y,0,360,100,100);//face
		setfillstyle(SOLID_FILL,14);
		fillellipse(x,y,100,100);

		ellipse(x-30,y-30,0,360,20,40);//eye
		setfillstyle(SOLID_FILL,random(6));
		fillellipse(x-30,y-30,20,40);
		ellipse(x+30,y-30,0,360,20,40);
		fillellipse(x+30,y-30,20,40);

		setcolor(RED);//mouth
		ellipse(x,y+10,180,0,60,50);
		delay(500);
	}
	closegraph();
}
```

iii. Write a program to draw a moving car using graphics function.

```
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <dos.h>
int main()
{
  int gd = DETECT, gm;
  int i, maxx, midy;
      /* initialize graphic mode */
  initgraph(&gd, &gm,
  "C:\\TurboC3\\BGI"); /* maximum
  pixel in horizontal axis */ maxx =
  getmaxx();
  /* mid pixel in vertical axis */
  midy = getmaxy()/2;

      for(i=0; i< maxx-150; i=i+5)
      {
            /* clears screen */
            cleardevice();

            /* draw a white road */
            setcolor(WHITE);
            line(0, midy + 37, maxx, midy + 37);

            /* Draw Car */
            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);
/* Draw Wheels */
circle(30 + i, midy + 25, 9);
circle(90 + i, midy + 25, 9);
floodfill(30 + i, midy + 25, BLUE);
floodfill(90 + i, midy + 25, BLUE);
```

```
            /* Add delay of 0.1 milli seconds */
            delay(100);
          if(i==485)
                i=0;
            else if (kbhit())
                 break;
        }
        getch();
        closegraph();
        return 0;
}
```