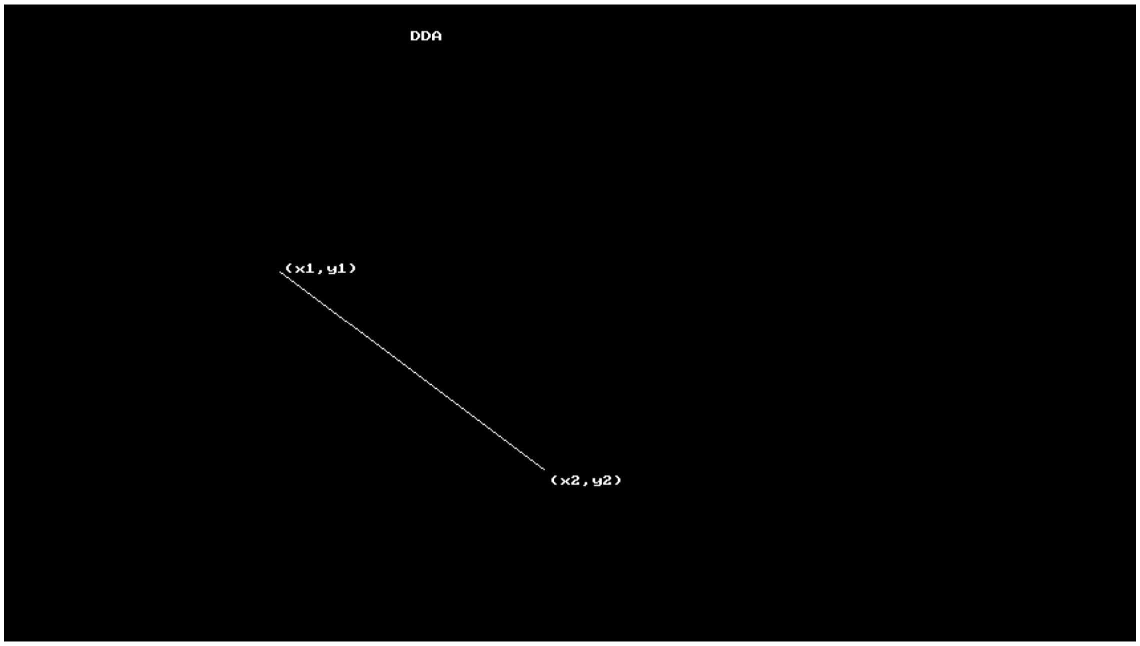
COMPUTER GRAPHICS LAB MANUAL

1. Implement DDA Algorithm for drawing a line segment b/w two given end points A(x1,y1) and B(x2,y2).

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
float round(float a);
void main()
{
	int gd=DETECT,gm;
	// gd=graphics driver (detects best graphics driver and assigns it as default, //gm=graphics mode.
	int x1,y1,x2,y2,steps,k;
	float xincr,yincr,x,y,dx,dy;
	printf("enter x1,y1");
	scanf("%d%d",&x1,&y1);
	printf("enter x2,y2");
	scanf("%d%d",&x2,&y2);
	initgraph(&gd,&gm,"c:\\TURBOC3\\BGI"); //initializes the graph
	dx=x2-x1;
	dy=y2-y1;
	if(abs(dx)>abs(dy))
	steps=abs(dx);
	else
	steps=abs(dy);
	xincr=dx/steps;
	yincr=dy/steps;
	x=x1;
	y=y1;
	for(k=1;k<=steps;k++)
	{
		delay(100); //for seeing the line drawing process slowly.
		x+=xincr;
		y+=yincr;
		putpixel(round(x),round(y),WHITE);
	}
	outtextxy(200,20,"DDA"); // for printing text at desired screen location.
	outtextxy(x1+5,y1-5,"(x1,y1)");
	outtextxy(x2+5,y2+5,"(x2,y2)");
	getch();
	closegraph(); // closes the graph and comes back to previous graphic mode.
}
float round(float a)
{
	int b=a+0.5;
	return b;
}
```

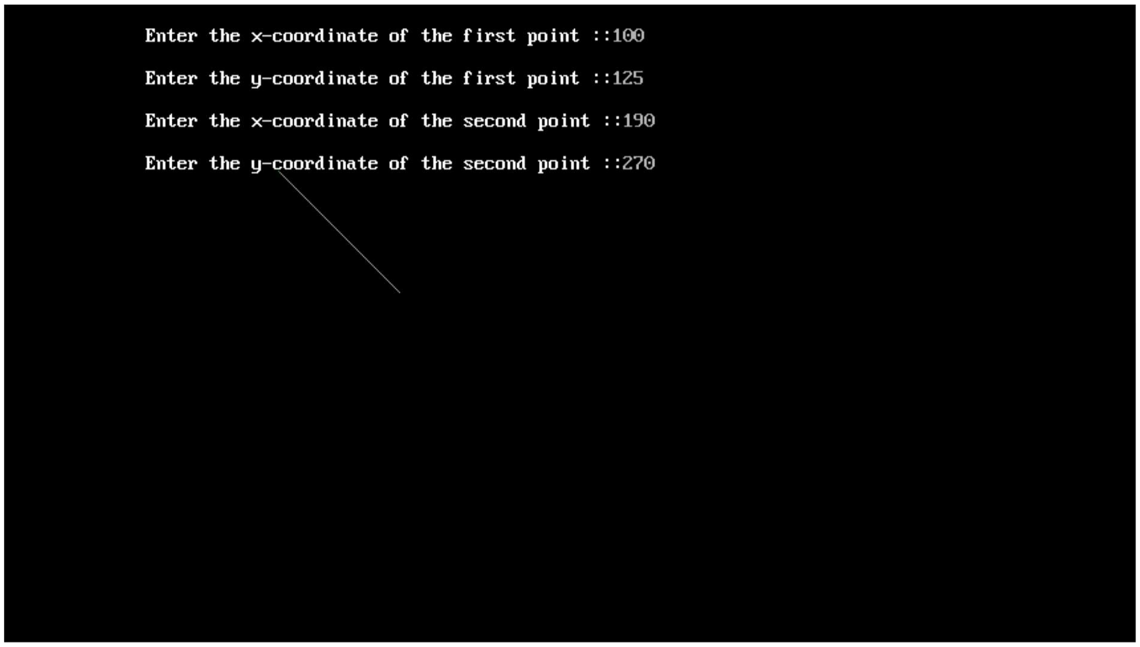
OUTPUT:



2. Write a C program for determining pixel activation line b/w two given points in order to draw line segment using Bresenham's line drawing algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        int x,y,x1,y1,x2,y2,p,dx,dy;
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
        printf("\nEnter the x-coordinate of the first point ::");
        scanf("%d",&x1);
        printf("\nEnter the y-coordinate of the first point ::");
        scanf("%d",&y1);
        printf("\nEnter the x-coordinate of the second point ::");
        scanf("%d",&x2);
        printf("\nEnter the y-coordinate of the second point ::");
        scanf("%d",&y2);
        x=x1;
        y=y1;
        dx=x2-x1;
        dy=y2-y1;
        putpixel(x,y,2);
        p=(2*dy-dx);
        while(x<=x2)
        {


                if(p<0)
                {
                        x=x+1;
                        p=p+2*dy;
                }
                else
                {
                        x=x+1;
                        y=y+1;
                        p=p+(2*dy)-(2*dx);
                }
                putpixel(x,y,7);
        }
        getch();
        closegraph();
}
```

<u>OUTPUT:</u>

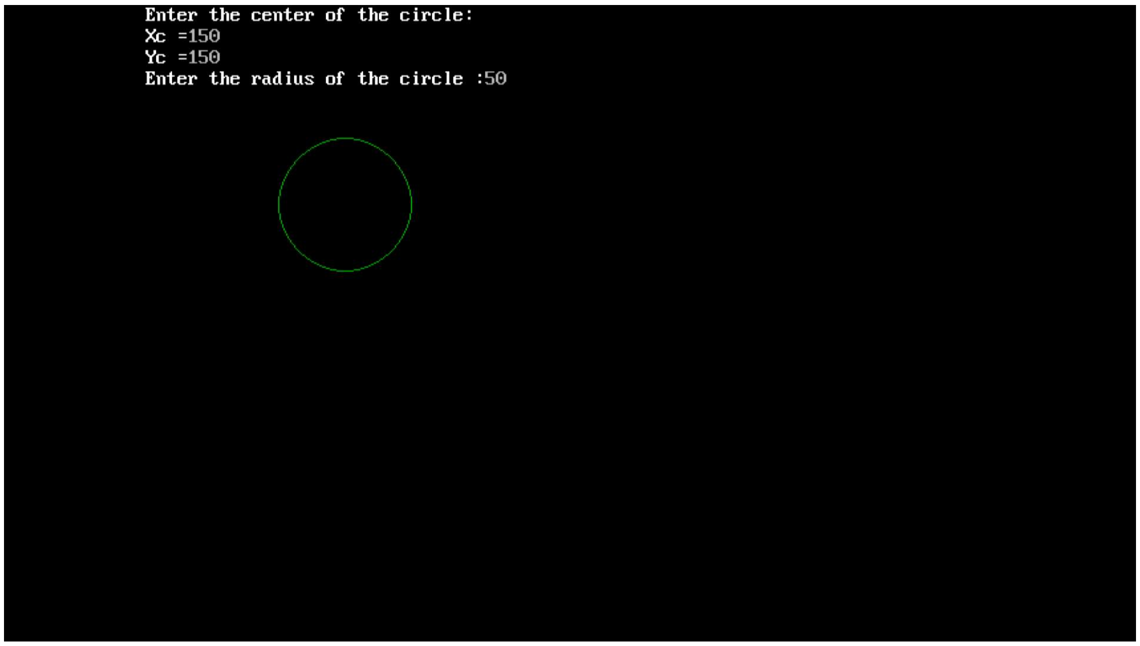


```
Enter the x-coordinate of the first point ::100

Enter the y-coordinate of the first point ::125

Enter the x-coordinate of the second point ::190

Enter the y-coordinate of the second point ::270
```

3. Implement mid-point circle generation algorithm for drawing a circle of given centre(x,y) & radius ‘r’.

```
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void draw_circle(int,int,int);
void symmetry(int,int,int,int);
void main()
{
        int xc,yc,R;
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
        printf("Enter the center of the circle:\n");
        printf("Xc =");
        scanf("%d",&xc);
        printf("Yc =");
        scanf("%d",&yc);
        printf("Enter the radius of the circle :");
        scanf("%d",&R);
        draw_circle(xc,yc,R);
        getch();
        closegraph();
}
void draw_circle(int xc,int yc,int rad)
{
        int x = 0;
        int y = rad;
        int p = 1-rad;
        symmetry(x,y,xc,yc);
        for(x=0;y>x;x++)
        {
                if(p<0)
                {
                        p += 2*x + 3;
                }
                else
                {
                        p += 2*(x-y) + 5;
                        y--;
                }
                symmetry(x,y,xc,yc);
                delay(50);
        }
}
void symmetry(int x,int y,int xc,int yc)
{
        putpixel(xc+x,yc-y,GREEN); //For pixel (x,y)
        delay(50);
        putpixel(xc+y,yc-x, GREEN); //For pixel (y,x)
        delay(50);
        putpixel(xc+y,yc+x, GREEN); //For pixel (y,-x)
        putpixel(xc+x,yc+y, GREEN); //For pixel (x,-y)
        delay(50);
        putpixel(xc-x,yc+y, GREEN); //For pixel (-x,-y)
        delay(50);
        putpixel(xc-y,yc+x, GREEN); //For pixel (-y,-x)
        delay(50);
        putpixel(xc-y,yc-x, GREEN); //For pixel (-y,x)
        delay(50);
        putpixel(xc-x,yc-y, GREEN); //For pixel (-x,y)
        delay(50);
}
```

OUTPUT:

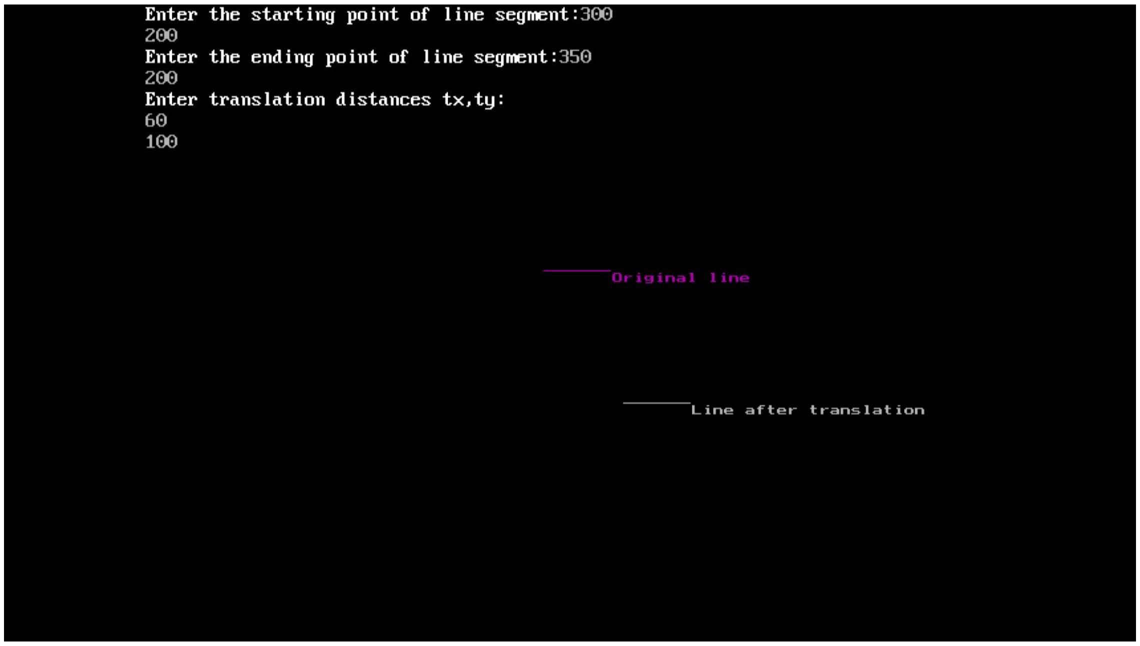


```
Enter the center of the circle:
Xc =150
Yc =150
Enter the radius of the circle :50
```

4. Design and develop a C program to implement translation of a line.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
        int gd=DETECT,gm;
        float x1,y1,x2,y2,sx,sy,x3,y3,x4,y4;
        initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
        printf("Enter the starting point coordinates:");
        scanf("%f %f",&x1,&y1);
        printf("Enter the ending point coordinates:");
        scanf("%f %f",&x2,&y2);
        printf("Enter scaling factors sx,sy:\n");
        scanf("%f%f",&sx,&sy);
        setcolor(5);
        line(x1,y1,x2,y2);
        outtextxy(x2+2,y2+2,"Original line");
        x3=x1*sx;
        y3=y1*sy;
        x4=x2*sx;
        y4=y2*sy;
        setcolor(7);
        line(x3,y3,x4,y4);
        outtextxy(x3+2,y3+2,"Line after scaling");
        getch();
}
```

OUTPUT:

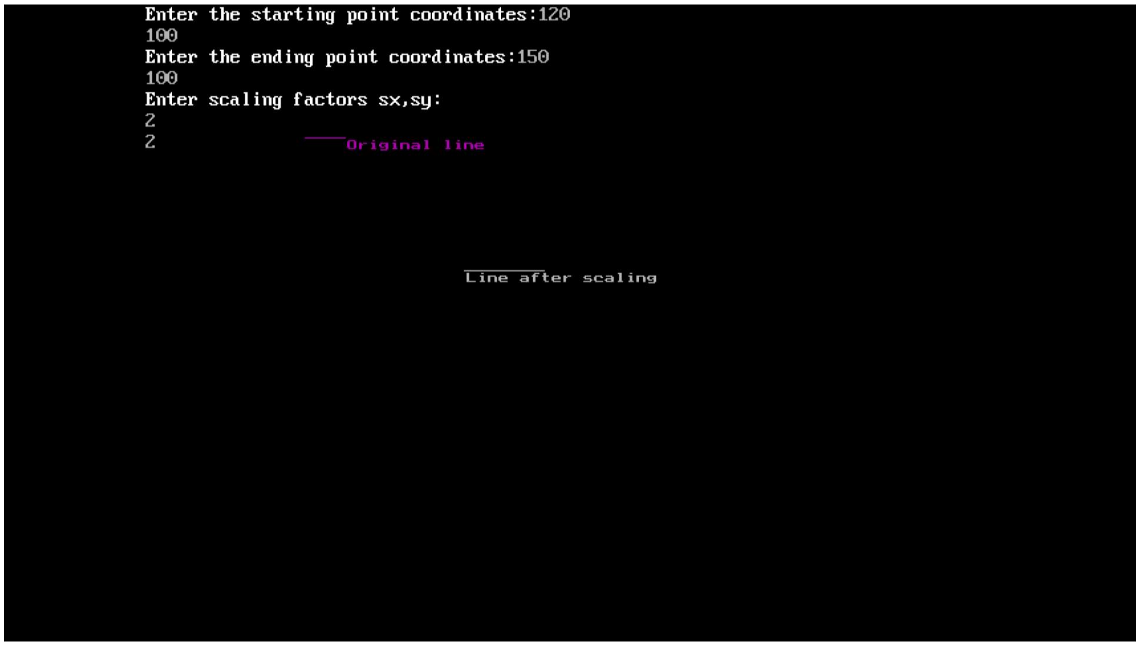


```
Enter the starting point of line segment:300
200
Enter the ending point of line segment:350
200
Enter translation distances tx,ty:
60
100
```

5. Design and develop a C program to implement scaling of a line.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
        int gd=DETECT,gm;
        float x1,y1,x2,y2,sx,sy,x3,y3,x4,y4;
        initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
        printf("Enter the starting point coordinates:");
        scanf("%f %f",&x1,&y1);
        printf("Enter the ending point coordinates:");
        scanf("%f %f",&x2,&y2);
        printf("Enter scaling factors sx,sy:\n");
        scanf("%f%f",&sx,&sy);
        setcolor(5);
        line(x1,y1,x2,y2);
        outtextxy(x2+2,y2+2,"Original line");
        x3=x1*sx;
        y3=y1*sy;
        x4=x2*sx;
        y4=y2*sy;
        setcolor(7);
        line(x3,y3,x4,y4);
        outtextxy(x3+2,y3+2,"Line after scaling");
        getch();
}
```

OUTPUT;

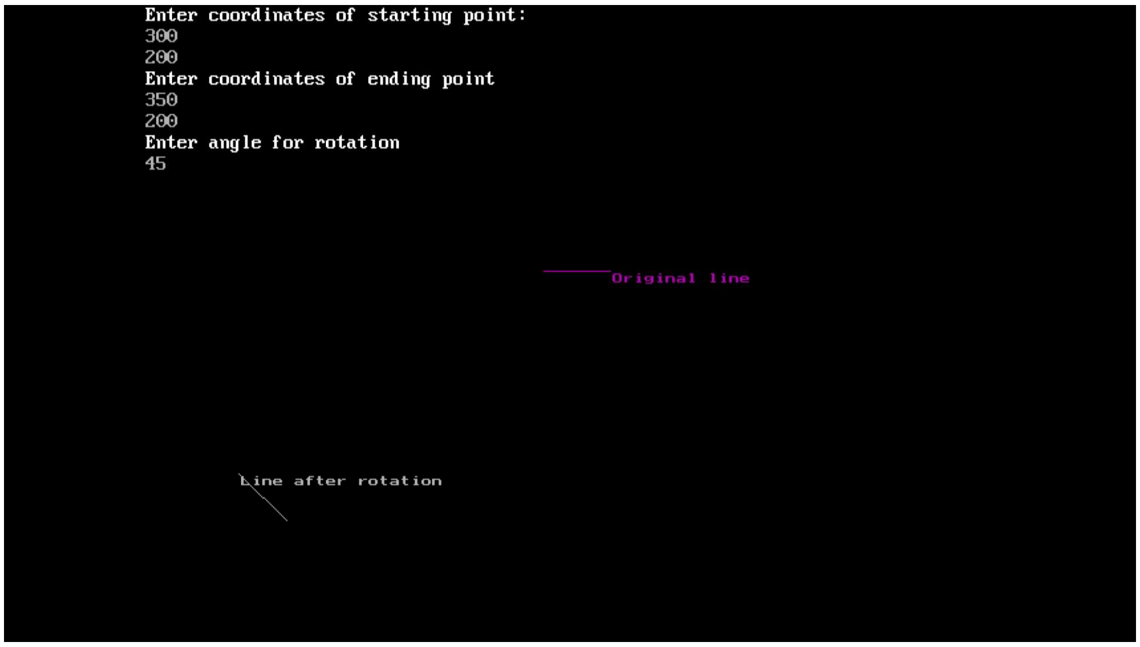


```
Enter the starting point coordinates:120
100
Enter the ending point coordinates:150
100
Enter scaling factors sx,sy:
2
2
```

6. Design and develop a C program to implement rotation of a line.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
	int gd=DETECT,gm;
	float x1,y1,x2,y2,x3,y3,x4,y4,a,t;
	initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
	printf("Enter coordinates of starting point:\n");
	scanf("%f%f",&x1,&y1);
	printf("Enter coordinates of ending point\n");
	scanf("%f%f",&x2,&y2);
	printf("Enter angle for rotation\n");
	scanf("%f",&a);
	setcolor(5);
	line(x1,y1,x2,y2);
	outtextxy(x2+2,y2+2,"Original line");
	t=a*(3.14/180);
	x3=(x1*cos(t))-(y1*sin(t));
	y3=(x1*sin(t))+(y1*cos(t));
	x4=(x2*cos(t))-(y2*sin(t));
	y4=(x2*sin(t))+(y2*cos(t));
	setcolor(7);
	line(x3,y3,x4,y4);
	outtextxy(x3+2,y3+2,"Line after rotation");
	getch();
}
```

OUTPUT;

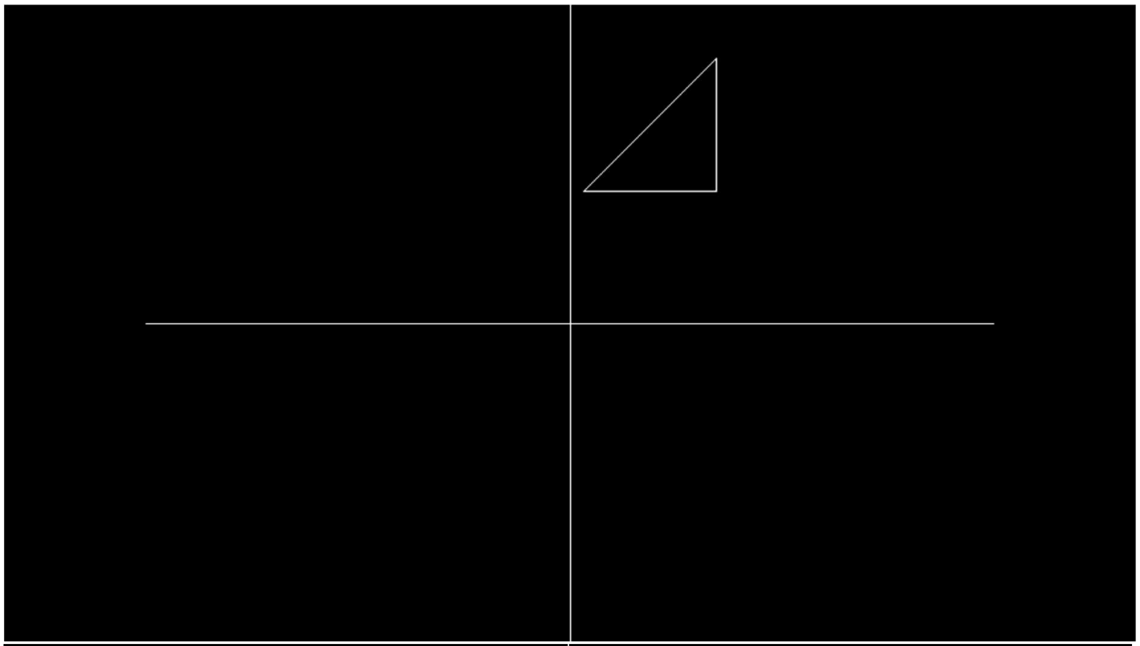


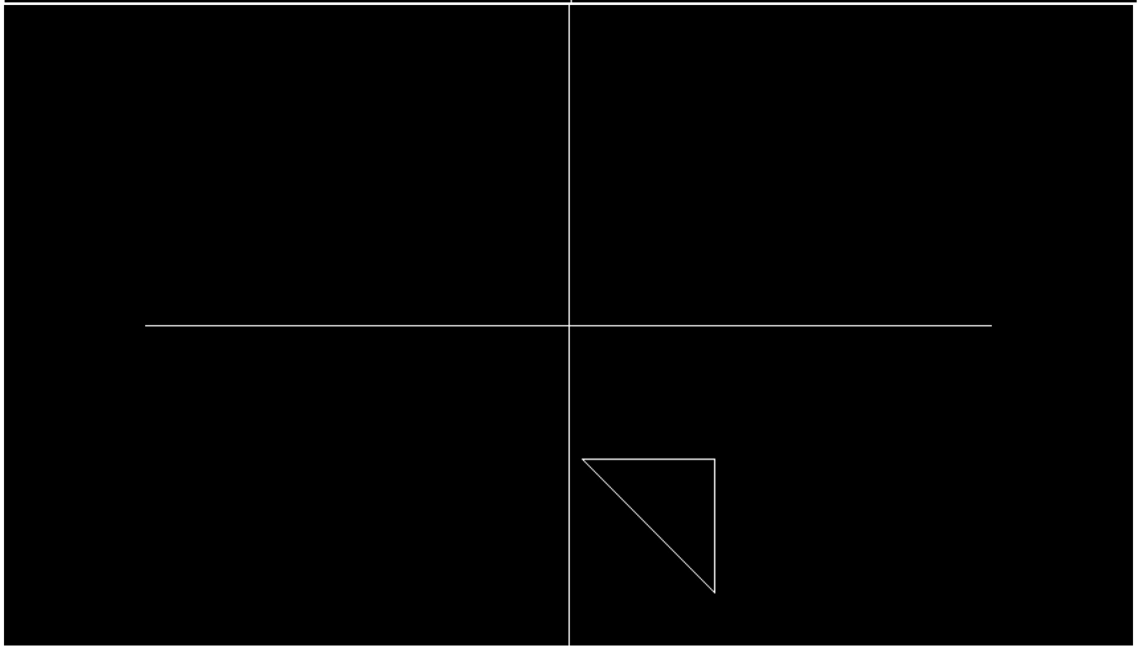
```
Enter coordinates of starting point:
300
200
Enter coordinates of ending point
350
200
Enter angle for rotation
45
```

7. Design and develop a C program to implement reflection along x-axis.

```
# include <stdio.h>
# include <conio.h>
# include <graphics.h>
# include <math.h>
char IncFlag;
int PolygonPoints[3][2] ={{10,100},{110,100},{110,200}};
void PolyLine()
{
        int iCnt;
        cleardevice();
        line(0,240,640,240);
        line(320,0,320,480);
        for (iCnt=0; iCnt<3; iCnt++)
        {
                line(PolygonPoints[iCnt][0],PolygonPoints[iCnt][1],
                PolygonPoints[(iCnt+1)%3][0],PolygonPoints[(iCnt+1)%3][1]);
        }
}
void Reflect()
{
        float Angle;
        int iCnt;
        int Tx,Ty;
        printf("endl");
        for (iCnt=0; iCnt<3; iCnt++)
        {
                PolygonPoints[iCnt][1] = (480 - PolygonPoints[iCnt][1]);
        }
}

void main()
{
        int gDriver = DETECT, gMode;
        int iCnt;
        initgraph(&gDriver, &gMode, "C:\\TurboC3\\BGI");
        for (iCnt=0; iCnt<3; iCnt++)
        {
                PolygonPoints[iCnt][0] += 320;
                PolygonPoints[iCnt][1] = 240 - PolygonPoints[iCnt][1];
        }
        PolyLine();
        getch();
        Reflect();
        PolyLine();
        getch();
}
```

OUTPUT:

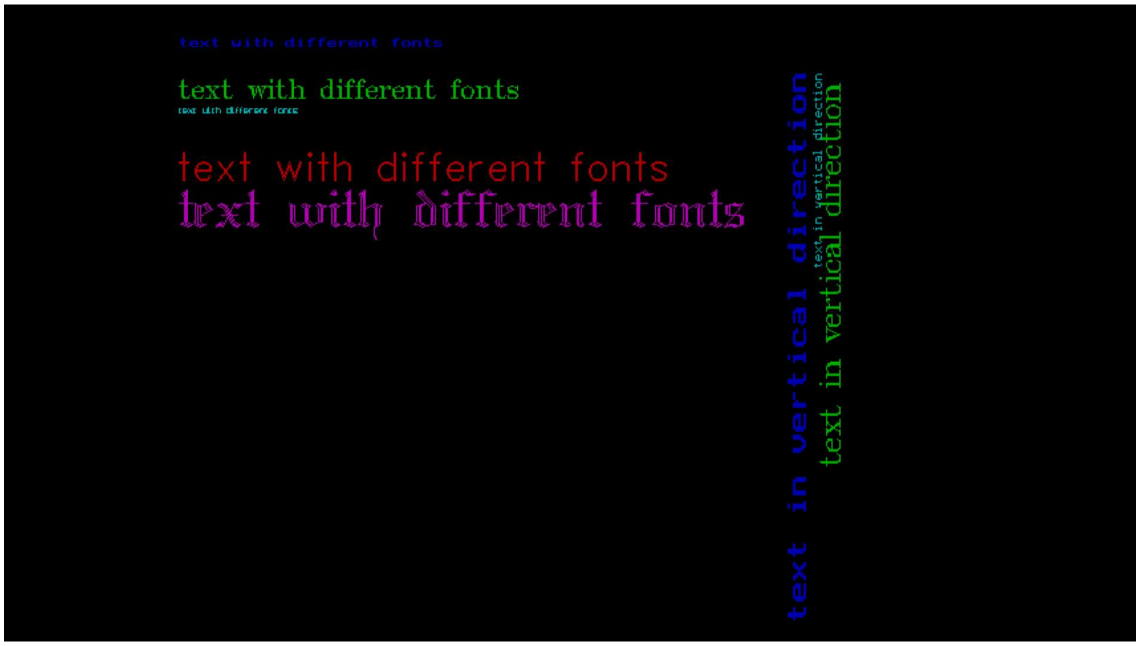




8. Creating various types of texts and fonts.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        int gd=DETECT,gm,x=25,y=25,font=10;
        initgraph(&gd,&gm,"C:\\turboC3\\BGI");
        for(font=0;font<=4;font++)
        {
                settextstyle(font,HORIZ_DIR,font+1); // sets font type, font direction, size
                setcolor(font+1); // sets color for text.
                outtextxy(x,y,"text with different fonts"); // prints message on screen at (x,y)
                y=y+25;
        }
        for(font=0;font<=2;font++)
        {
                settextstyle(font,VERT_DIR,font+2);
                setcolor(font+1);
                x=500;
                y=50;
                outtextxy(x,y,"text in vertical direction");
                y=y+25;
        }
        getch();
        closegraph();
}
```

OUTPUT:

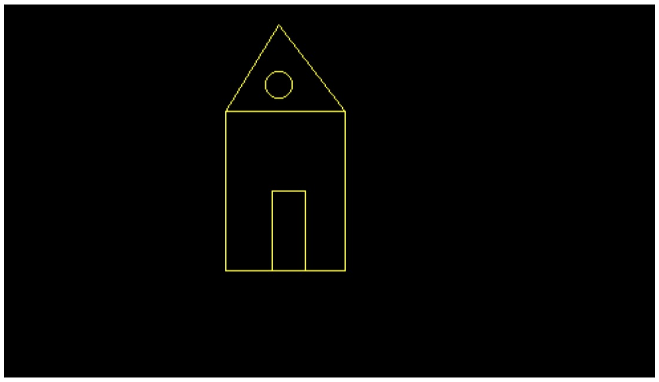


```
text with different fonts
text with different fonts
text with different fonts
text with different fonts
text with different fonts

text in vertical direction
text in vertical direction
text in vertical direction
```

9. Creating 2-D House object.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
        setcolor(5);
        rectangle(60,80,150,200);
        rectangle(95,140,120,200);
        line(60,80,100,15);
        line(100,15,150,80);
        circle(100,60,10);
        getch();
        closegraph();
}
```

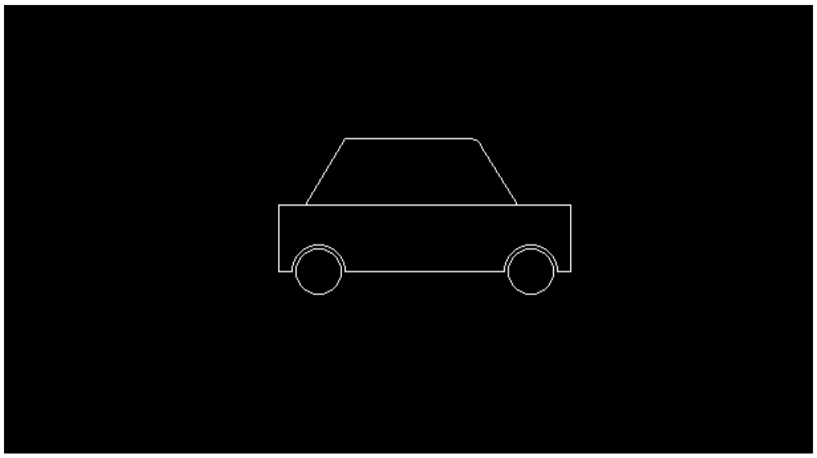
OUTPUT:



10. Creating 2-D Car object.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
        cleardevice();
        line( 150, 100, 242, 100);
        ellipse(242, 105, 0, 90, 10, 5);
        line(150, 100, 120, 150);
        line(252, 105, 280, 150);
        line(100, 150, 320, 150);
        line(100, 150, 100, 200);
        line(320, 150, 320, 200);
        line(100, 200, 110, 200);
        line(320, 200, 310, 200);
        arc(130, 200, 0, 180, 20);
        arc( 290, 200, 0, 180, 20);
        line( 270, 200, 150, 200);
        circle(130, 200, 17);
        circle(290, 200, 17);
        getch();
}
```

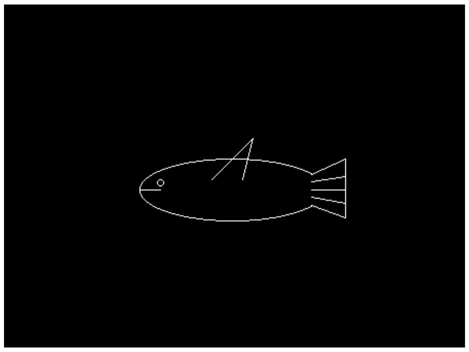
OUTPUT:



11. Creating a 2-D Fish object.

```
#include<stdlib.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>
#include<ctype.h>
void main()
{
	int gd=DETECT,gm;
	initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
	cleardevice();
	ellipse(520,200,30,330,90,30);
	circle(450,193,3);
	line(430,200,450,200);
	line(597,185,630,170);
	line(597,215,630,227);
	line(630,170,630,227);
	line(597,200,630,200);
	line(597,192,630,187);
	line(597,207,630,213);
	line(500,190,540,150);
	line(530,190,540,150);
	getch();
}
```

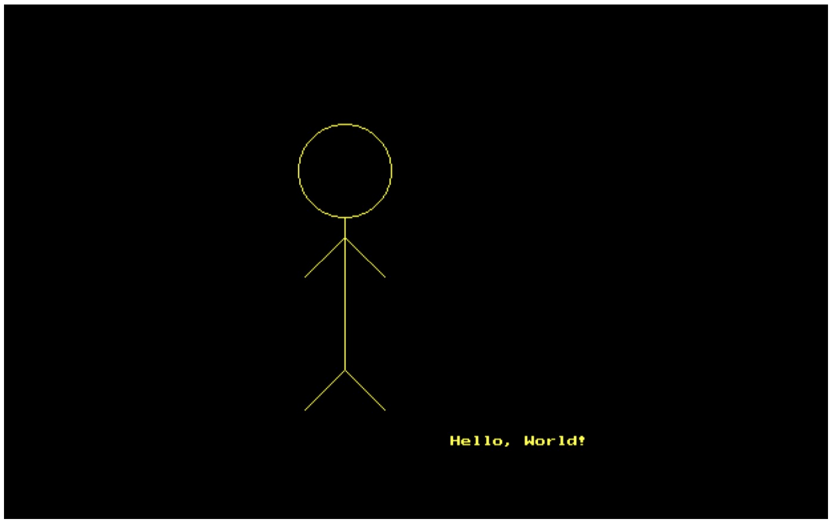
OUTPUT:



12. Creating 2-D Shape of a man.

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
void main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
setcolor(9);
circle(150,150,35);
line(150,185,150,300);
line(150,200,120,230);
line(150,200,180,230);
line(150,300,120,330);
line(150,300,180,330);
outtextxy(230,350,"HI, This is Computer Graphics");
getch();
}
```

<u>OUTPUT:</u>



```
Hello, World!
```