Name- Devesh Kumar Sharma

Reg no.- 2018105172

Graphics LAB ASSIGNMENTS - 1 to 15

Date – 26/10/2021

Submitted to – Mr. Napoleon

LAB ASSIGNMENT 1

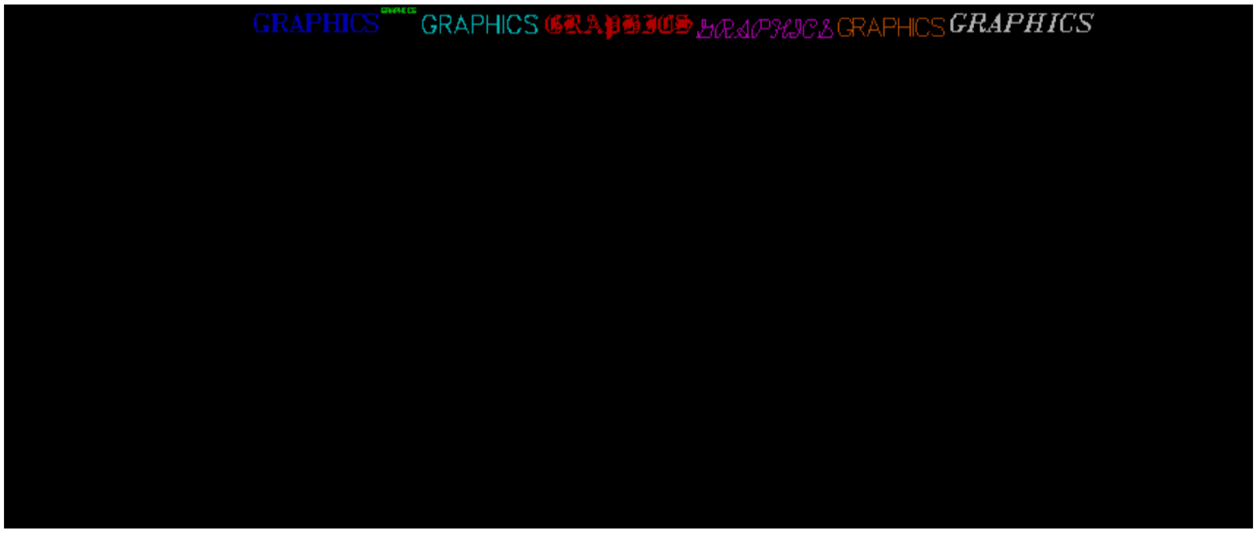
Q1- Write a C program to print the word "Graphics" in 10 different fonts and colors.

Sol-

Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
int main(){
int i;
int gd=DETECT, gm;
initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
for(i=0; i<10;i++){
setcolor(i);
settextstyle(i,0,1);
outtext("GRAPHICS");
}
getch();
closegraph();
return 0;
}
```

Output-



```
GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS
```

LAB ASSIGNMENT 2

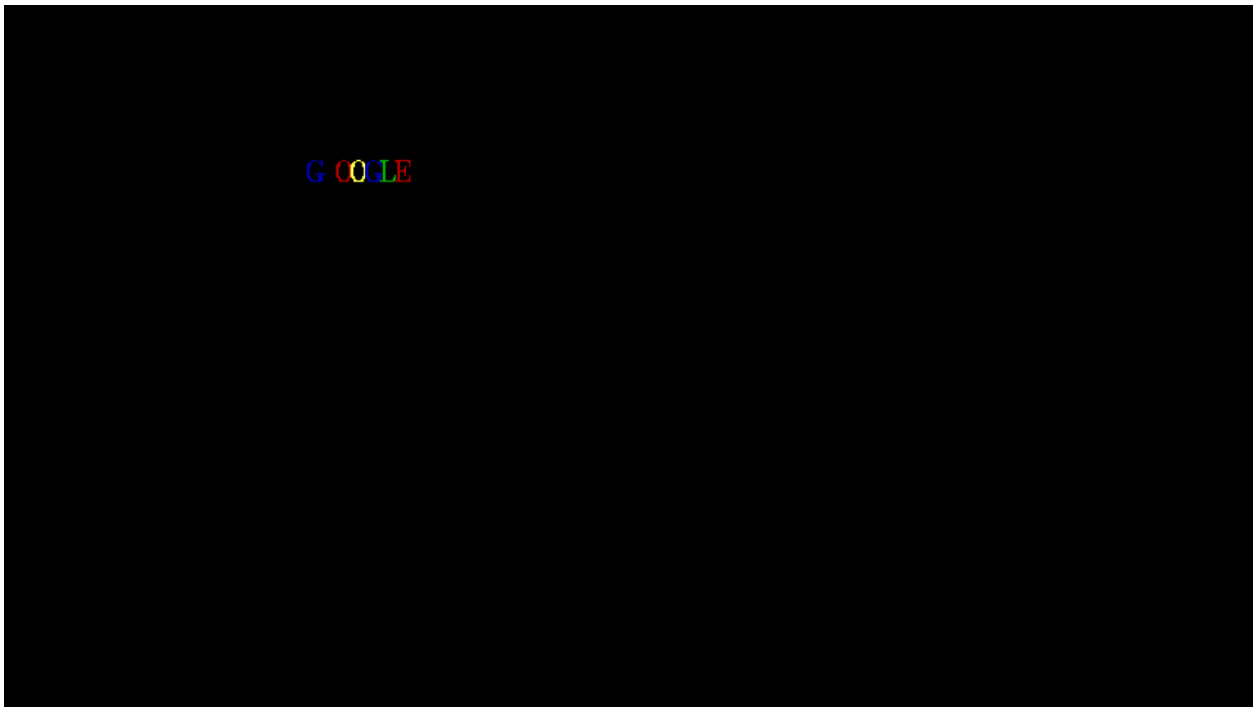
Q2- Write a C program to print the word “GOOGLE” with its default colors.

Code-

```
#include<stdio.h>

#include<conio.h>

#include<dos.h>

#include<graphics.h>


void main(){

	int gdriver = DETECT,gmode,i;

	initgraph(&gdriver,&gmode,"C:\\Turboc3\\BGI");

	settextstyle(1,0,8);

	setcolor(BLUE);

	outtext("G");

	setcolor(RED);

	outtext("O");

	setcolor(YELLOW);

	outtext("O");

	setcolor(BLUE);

	outtext("G");
```

```
        setcolor(GREEN);

        outtext("L");

        setcolor(RED);

        outtext("E");

        getch();

}
```

Output-



```
G OOGLE
```

LAB ASSIGNMENT 3

Q3- Write a C program to print the word text away from different directions(left, right, top, bottom).

Sol-

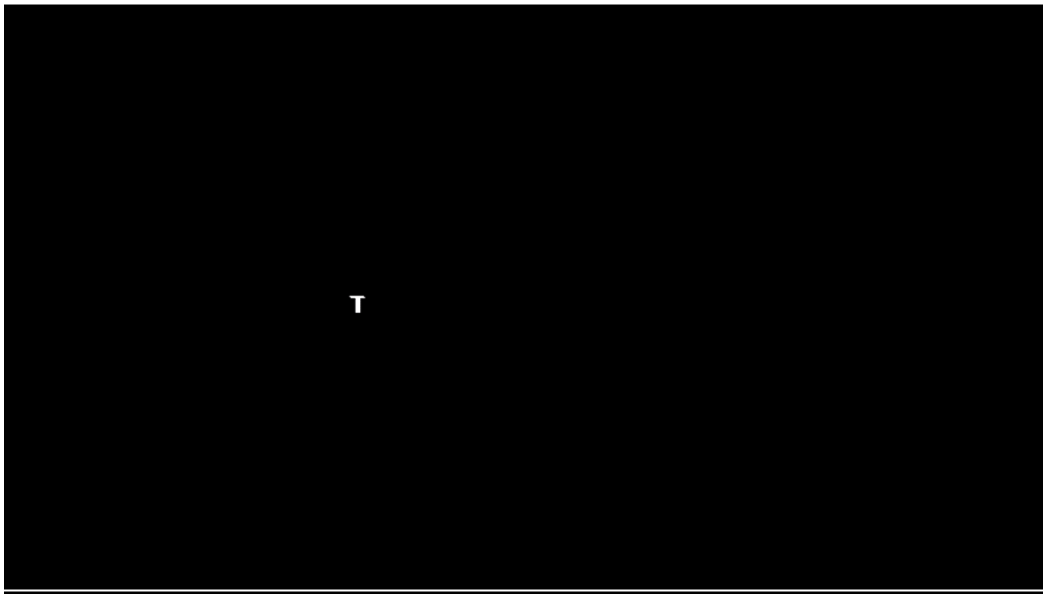
Code-

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<stdlib.h>x

int main(){
```

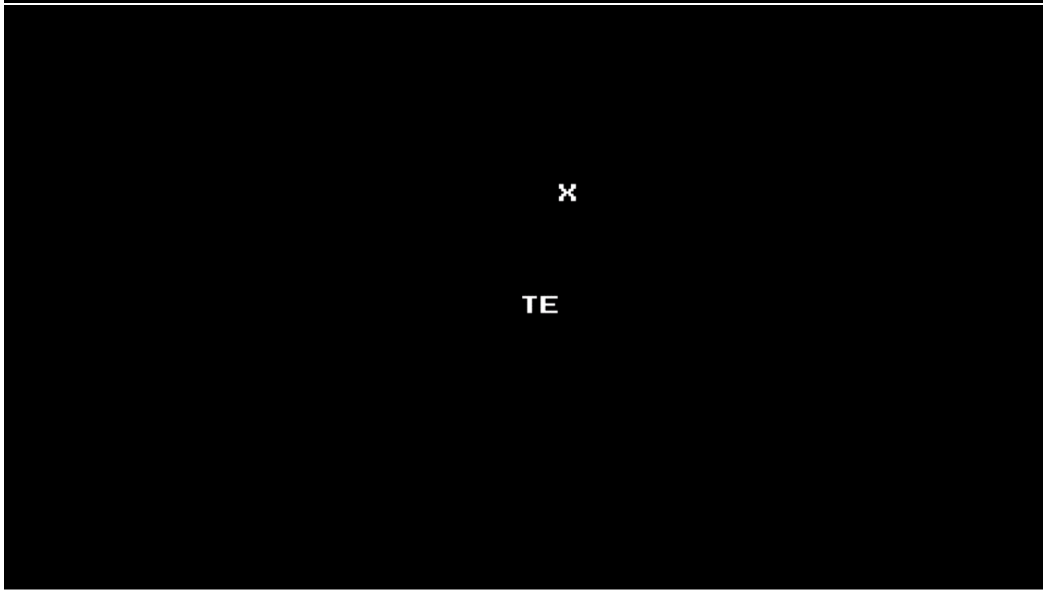
```
int i, gd=DETECT, gm;
initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
settextstyle(DEFAULT_FONT, HORIZ_DIR, 2);
for(i=0; i<=getmaxx()/2;i++){
outtextxy(i, getmaxy()/2, "T");
delay(5);
cleardevice();
}

for(i=getmaxx();i>=getmaxx()/2+15;i--){
outtextxy(getmaxx()/2, getmaxy()/2, "T");
outtextxy(i, getmaxy()/2, "E");
delay(5);
cleardevice();
}
for(i=0;i<=getmaxy()/2;i++){
outtextxy(getmaxx()/2, getmaxy()/2, "T");
outtextxy(getmaxx()/2+15, getmaxy()/2, "E");
outtextxy(getmaxx()/2+30, i, "X");
delay(5);
cleardevice();
}
for(i=getmaxy(); i>=getmaxy()/2; i--){
outtextxy(getmaxx()/2, getmaxy()/2, "T");
outtextxy(getmaxx()/2+15, getmaxy()/2, "E");
outtextxy(getmaxx()/2+30, getmaxy()/2, "X");
outtextxy(getmaxx()/2+45, i, "T");
delay(5);
if(i==getmaxy()/2)
break;
cleardevice();
}
getch();
closegraph();
return 0;
}
```

Output-

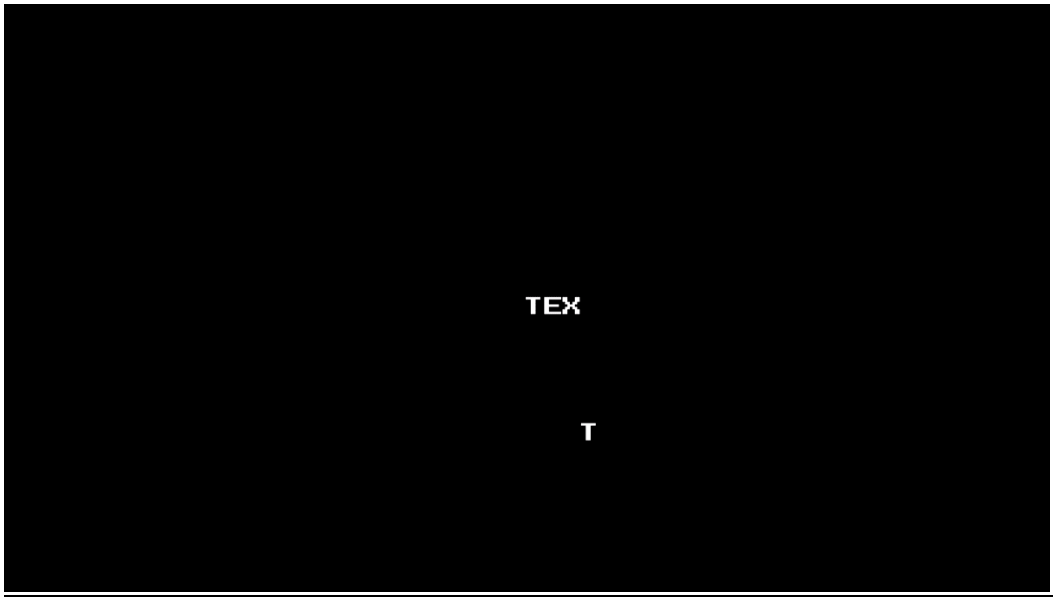
T is coming from left E from right X from top and T from bottom.



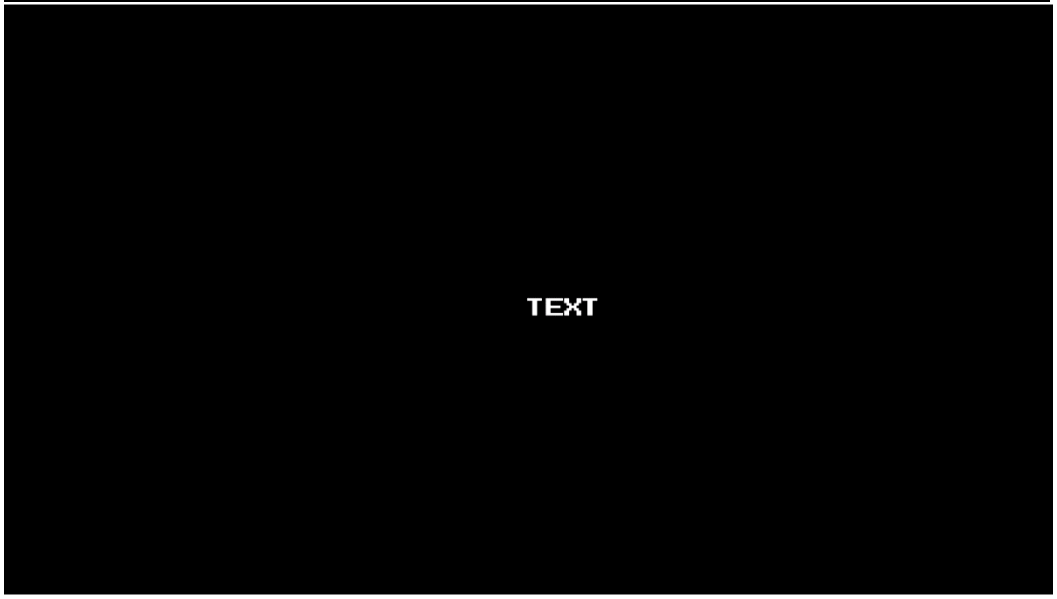
```
T
```



```
X
TE
```



```
TEX

        T
```



```
TEXT
```

LAB ASSIGNMENT 4

Q4- Write a C program to print the figures.

Sol-

Figure 1 Code-

```
#include<stdio.h>

#include<conio.h>

#include<dos.h>

#include<graphics.h>
```

```
void main(){
	int gd = DETECT,gm,i;
	initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
	rectangle(50,100,250,200);
	line(50,250,250,250);
	circle(150,350,50);
	getch();
}
```

Figure 2 Code-

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

void main(){
	int gd = DETECT,gm,i;
	initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
	rectangle(100,100,300,300);
	line(100,100,300,300);
	line(100,300,300,100);
	getch();
}
```

Figure 3 Code-

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

```
void main(){
	int gd = DETECT,gm,i;
	initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
	rectangle(200,200,300,300);
	line(200,200,250,100);
	line(300,200,250,100);
	line(300,200,400,250);
	line(300,300,400,250);
	line(300,300,250,400);
	line(200,300,250,400);
	line(200,300,100,250);
	line(100,250,200,200);
	getch();
}
```

Output-

Figure 1-

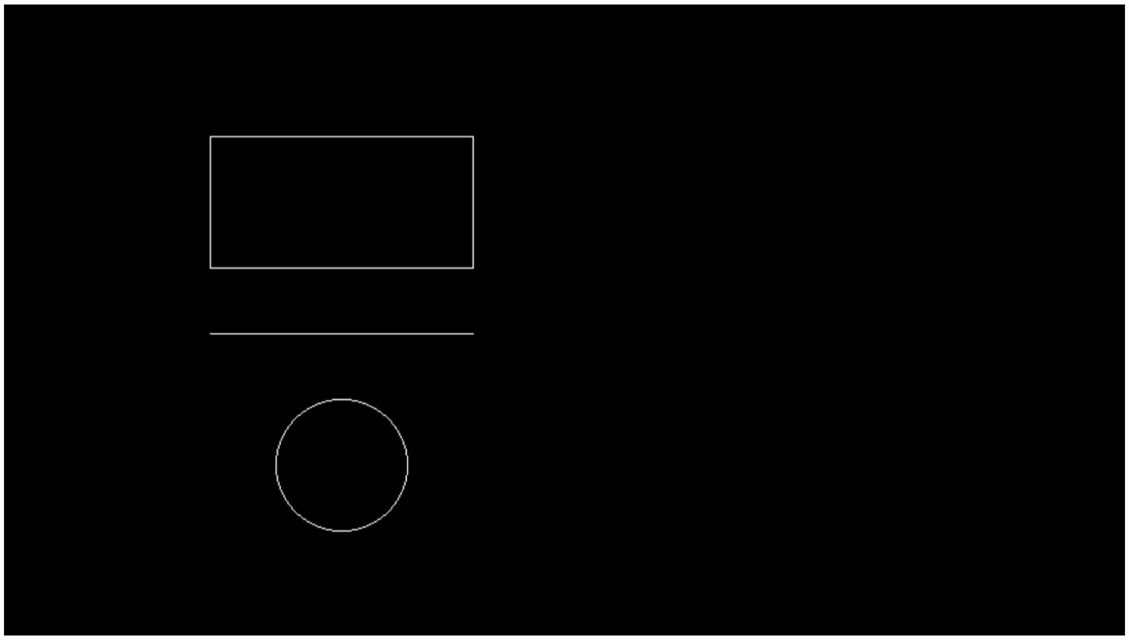
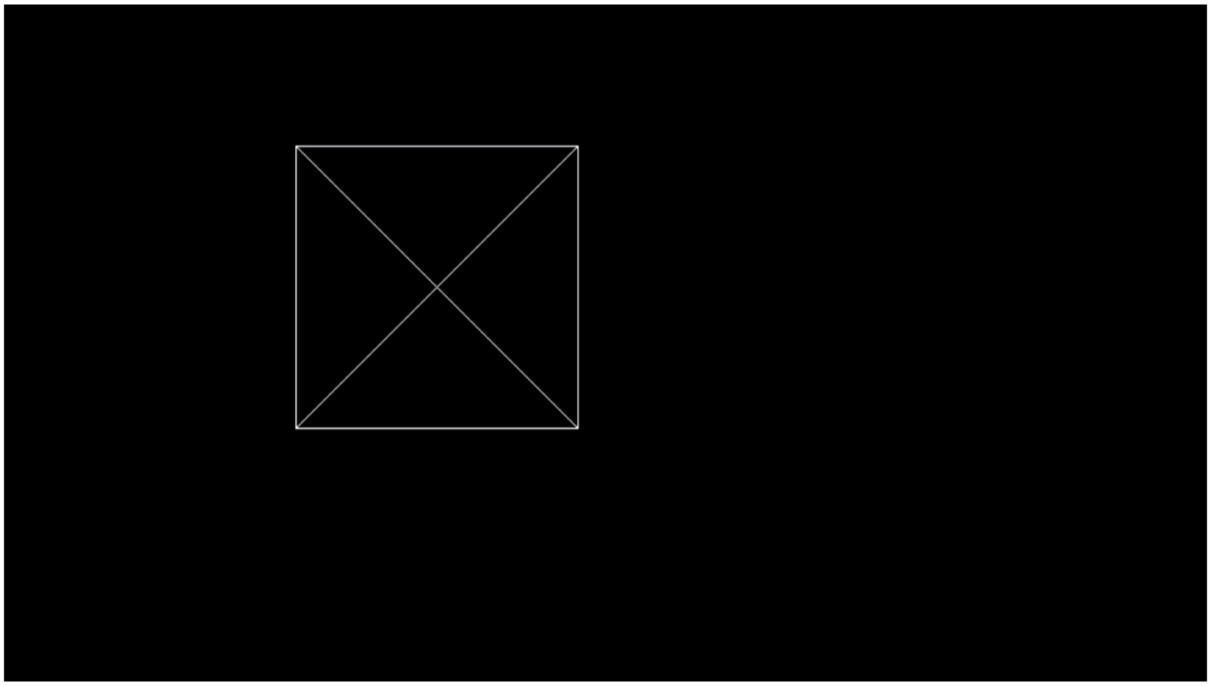


Figure 2-



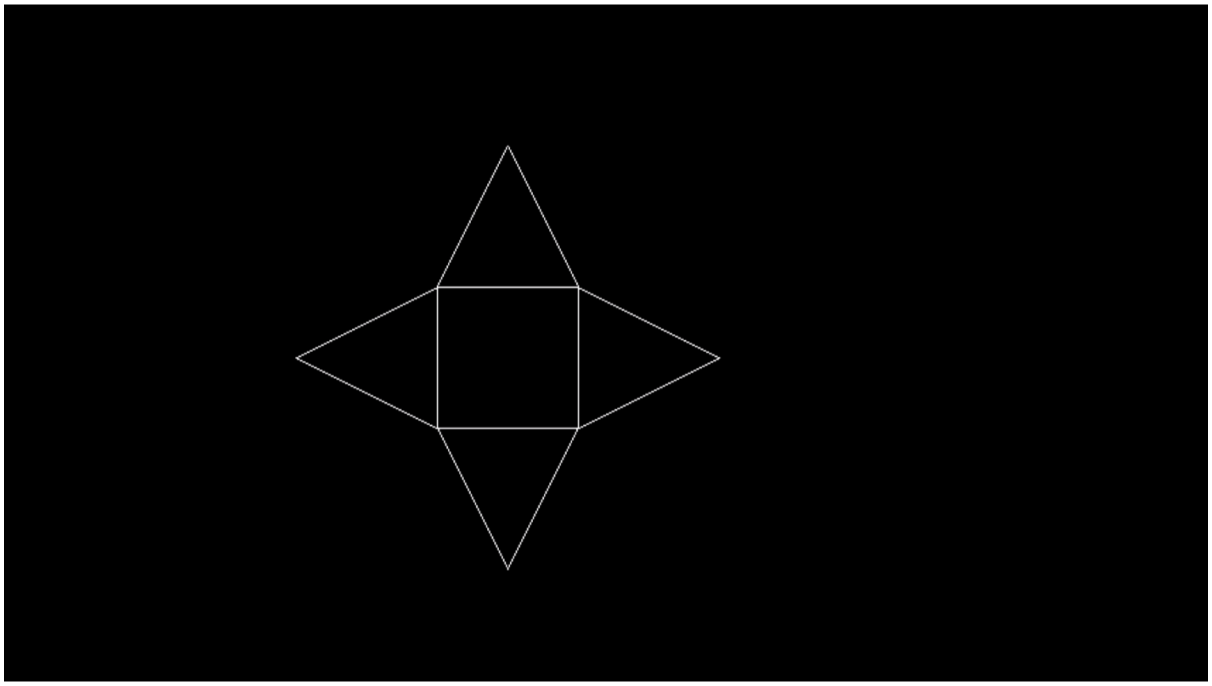


Figure 3-

LAB ASSIGNMENT 5

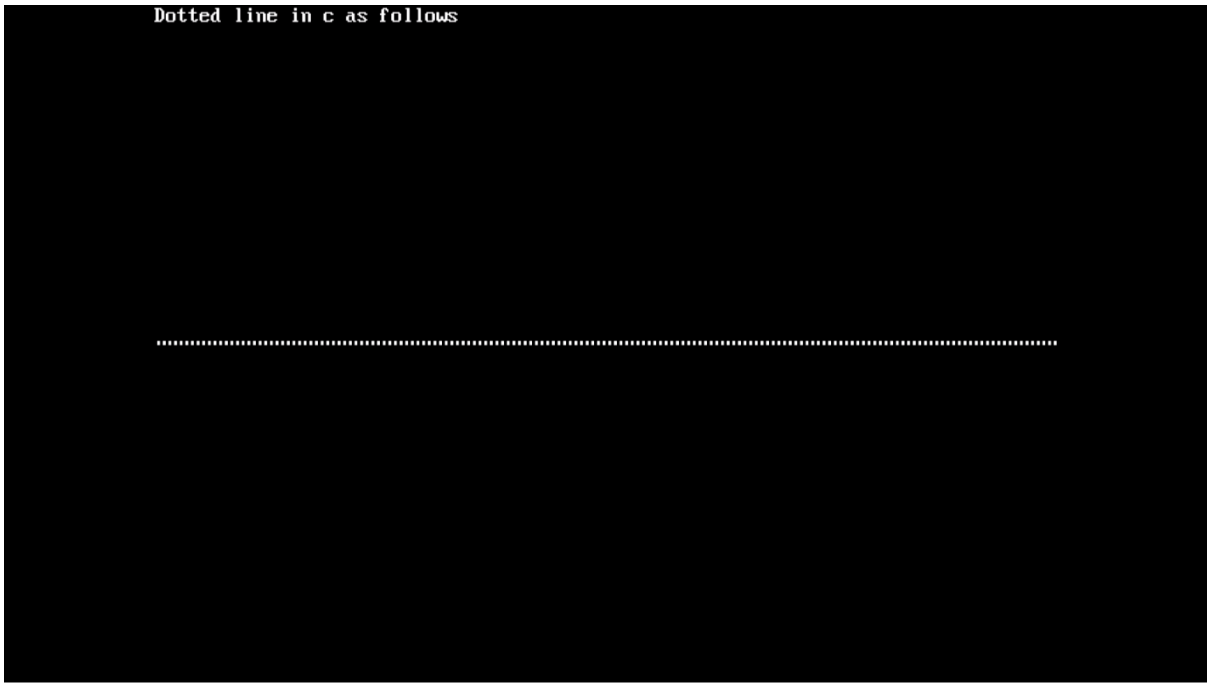
Q5- Write a C program to print a dotted line.

Sol-

Code-

```
#include<stdio.h>

#include<conio.h>

#include<dos.h>

#include<graphics.h>


void main(){

        int gd = DETECT,gm,i;

        initgraph(&gd,&gm,"C:\\Turboc3\\BGI");

        printf("Dotted line in c as follows\n");

        setlinestyle(1,1,3);

        line(0, getmaxy()/2, getmaxx(), getmaxy()/2);

        getch();

        closegraph();

}
```

Output-



```
Dotted line in c as follows
```

LAB ASSIGNMENT 6

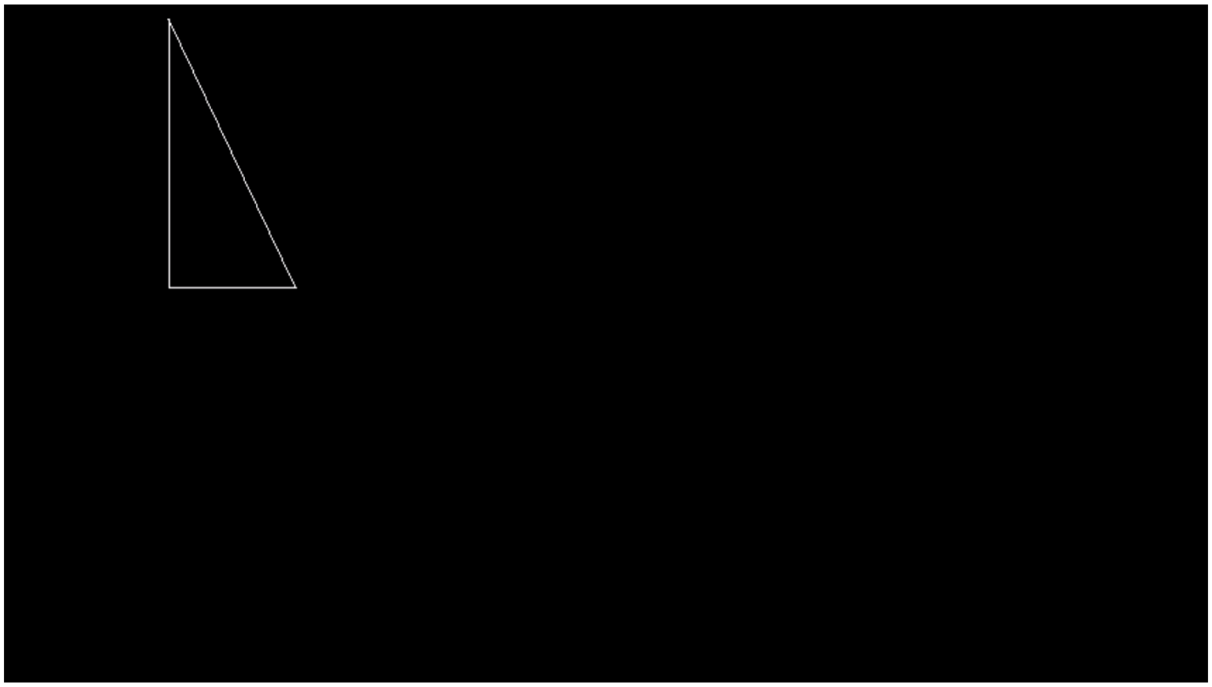
Q6- Print a triangle using DDA Algorithm whose vertex are predefined.

Sol-

Code-

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>


int abs(int n){

        return ((n > 0) ? n : n * (-1));

        }


void dda(int x0,int y0, int x1, int y1){

        int dx = x1 - x0;

        int dy = y1 - y0;

        int  i = 0;
        int step = abs(dx) > abs(dy) ? abs(dx) : abs(dy);

                float xInc = dx / (float) step;
                float yInc = dy / (float) step;

                float x = x0;
                float y = y0;
                for(i= 0; i <= step ; i++){
                        putpixel((int) x,(int) y, WHITE);
                        x += xInc;
                        y += yInc;
                        }
                }
void main(){
        int gd = DETECT,gm;
        initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
        dda(10,10,10,200);
        dda(10,200,100,200);
        dda(100,200,10,10);
        getch();
        closegraph();
}
```

Output-



LAB ASSIGNMENT 7

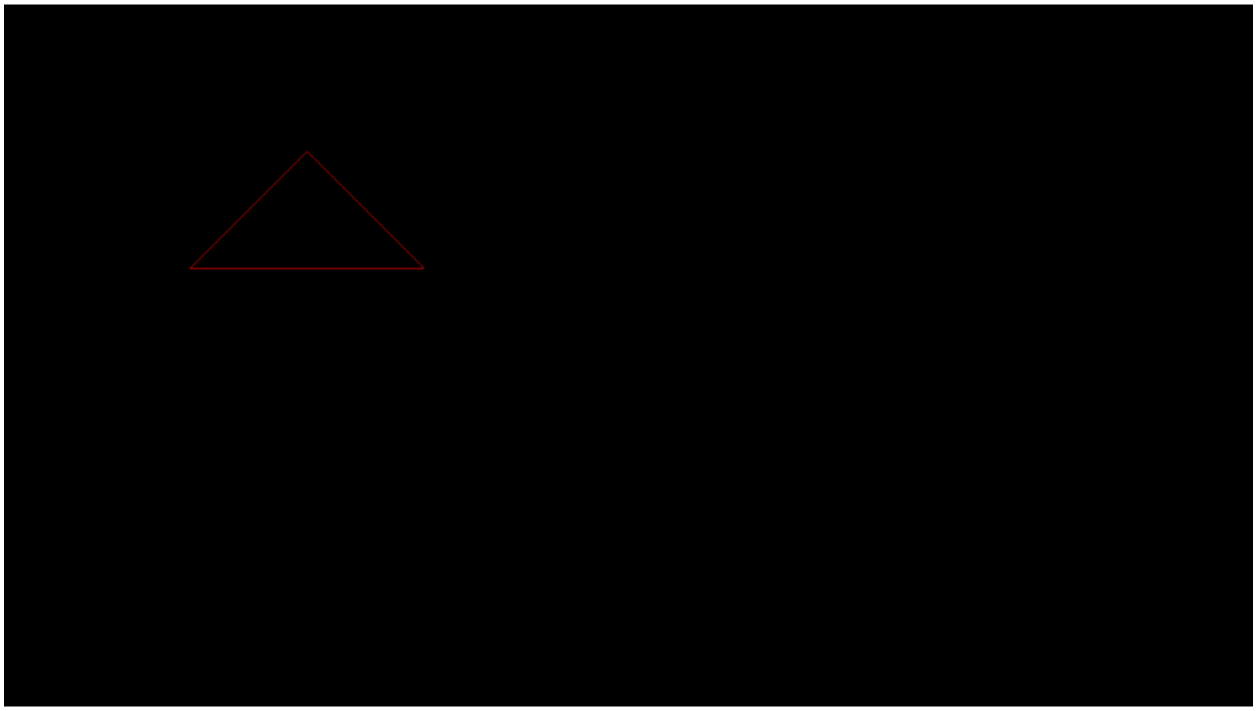
Q7- Print a triangle using Bresenhams Line drawing algorithm.

Sol-

Code-

```
#include<stdio.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>


void plotLineHigh(int x0,int y0,int x1,int y1)

{

   int dx,dy,xi,D,x,y;


   dx = x1 - x0;

   dy = y1 - y0 ;

   xi = 1;

   if (dx < 0){
          xi = -1;
          dx = -dx;
    }
    D = (2 * dx) - dy;
    x = x0;

     y=y0 ;
     while(y!=y1){
          putpixel(x, y,RED);
          if (D > 0) {
              x = x + xi;
              D = D + (2 * (dx - dy));
          }
          else
              D = D + 2*dx ;
       y+=1;
     }
}
void plotLineLow(int x0,int y0,int x1,int y1)
{
    int dx,dy,x,y,yi,D;

    dx = x1 - x0 ;
    dy = y1 - y0 ;
    yi = 1;
    if (dy < 0){
          yi = -1;
          dy = -dy;
    }
    D = (2 * dy) - dx;
    y = y0;

    x= x0 ;
    while(x!=x1){
           putpixel(x, y,RED);
           if (D > 0){
               y = y + yi ;
               D = D + (2 * (dy - dx));
           }
           else
               D = D + 2*dy ;
           x+=1;
    }
}

void plotLine(int x0,int  y0,int x1,int y1)
{
    if (abs(y1 - y0) < abs(x1 - x0) ) {
      if (x0 > x1) {
               plotLineLow(x1, y1, x0, y0);
      }
           else
               plotLineLow(x0, y0, x1, y1) ;

      }



    else if (y0 > y1) {
            plotLineHigh(x1, y1, x0, y0);
        }
        else
            plotLineHigh(x0, y0, x1, y1) ;



}
void main(){

  int gdriver=DETECT, gmode;
  initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");

  plotLine(100,100,20,180);
  plotLine(20,180,180,180);
  plotLine(180,180,100,100);
  getch();
}
```

Output-



LAB ASSIGNMENT 8

Q8- Perform the rotation of a square whose vertex are pre-defined.

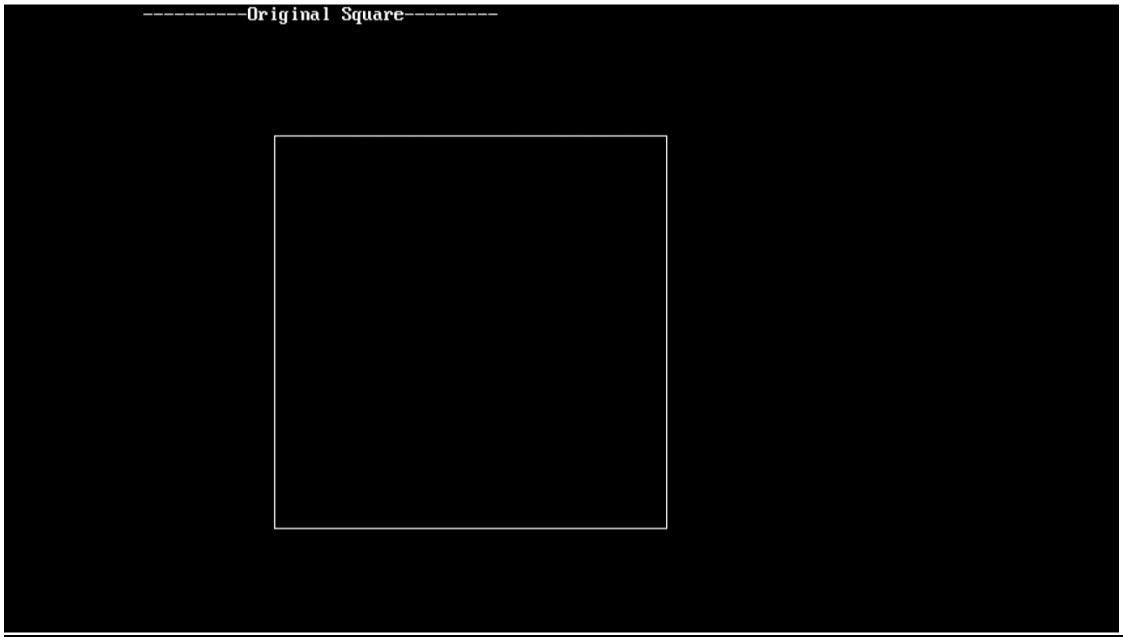
Sol-

Code-

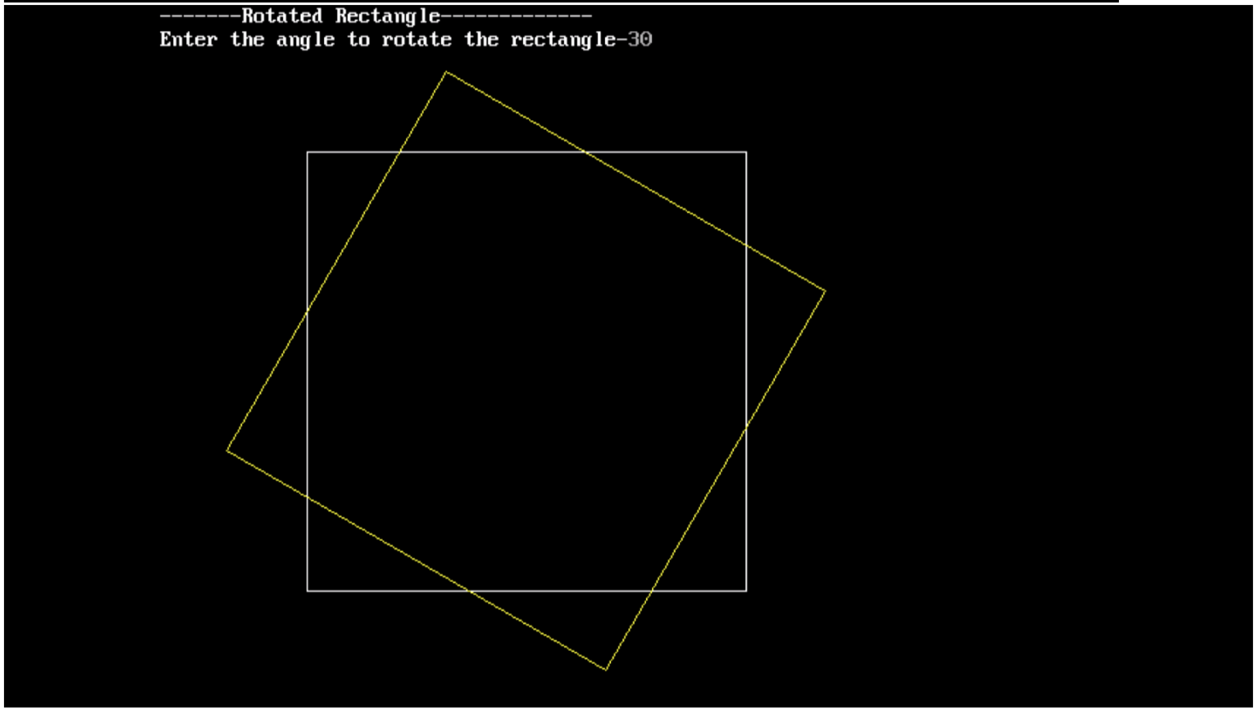
```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
double points[8];
void rectangleRotate(int cx, int cy, int w, int h, int angle)
{	int i = 0;
	double theta = (double)(angle%180)*M_PI/180.0;
	double dx = w/2;
	double dy = h/2;
	points[0]=(-dx*cos(theta) - dy*sin(theta) + cx);
	points[1]=(-dx*sin(theta) + dy*cos(theta) + cy);
	points[2]=(dx*cos(theta) - dy*sin(theta) + cx);
	points[3]=(dx*sin(theta) + dy*cos(theta) + cy);
	points[4]=(dx*cos(theta) + dy*sin(theta) + cx);
	points[5]=(dx*sin(theta) - dy*cos(theta) + cy);
	points[6]=(-dx*cos(theta) + dy*sin(theta) +cx);
	points[7]=(-dx*sin(theta) - dy*cos(theta) + cy);
	for(i=0; i<8; i+=2)
		line(points[i], points[(i+1)], points[(i+2)%8], points[(i+3)%8]);
}

int main(){
	int gd=DETECT, gm, angle;
	int x1=250, y1=250, x2=300, y2=300;
	initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
	printf("----------Original Square---------");
	rectangleRotate(x1, y1, x2, y2, 0);
	getch();
	clrscr();
	cleardevice();
	rectangleRotate(x1, y1, x2, y2, 0);
	printf("-------Rotated Rectangle-------------\n");
	printf("Enter the angle to rotate the rectangle-");
	scanf("%d",&angle);
	setcolor(YELLOW);
	rectangleRotate(x1, y1, x2, y2, angle);
	getch();
	closegraph();
	return 0;
```

}

Output-



```
----------Original Square---------
```



```
-------Rotated Rectangle------------
Enter the angle to rotate the rectangle-30
```

LAB ASSIGNMENT 9

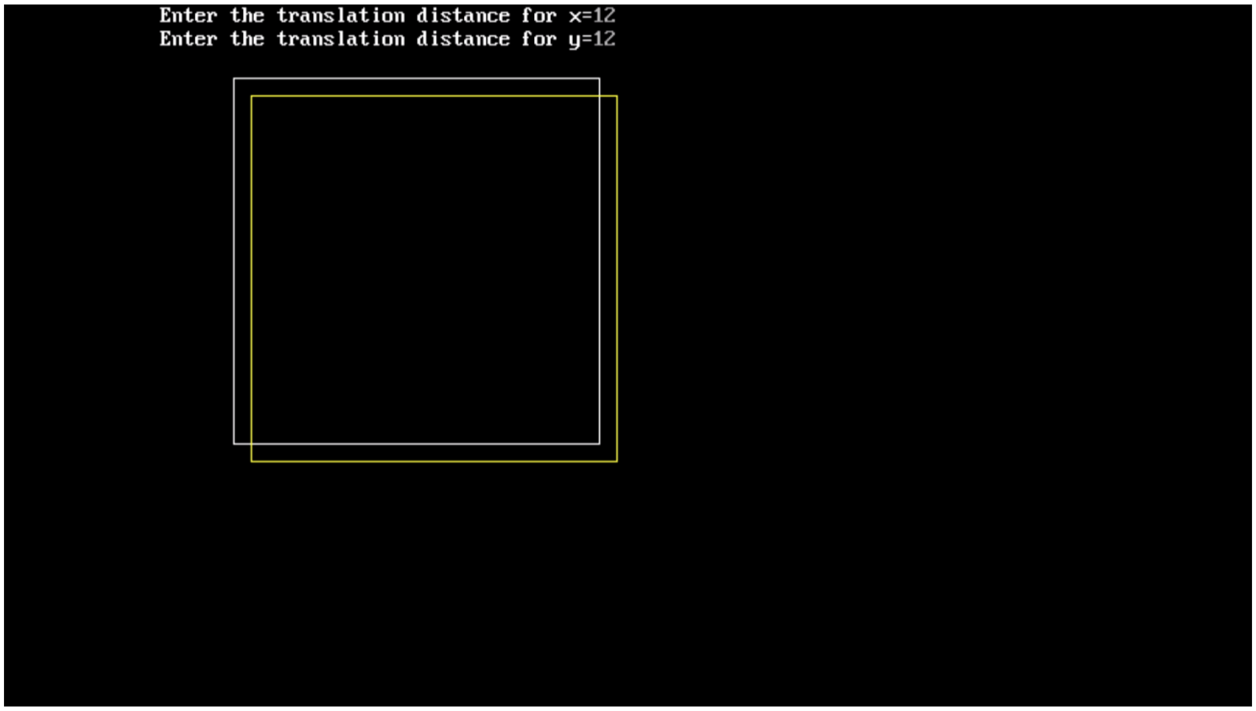
Q9- Perform translation of a square whose vertex are pre-defined.

Sol-

Code-

```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>


void main(){

        int gd=DETECT, gm, tx, ty;

        initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

        rectangle(50, 50, 300, 300);

        setcolor(14);

        printf("Enter the translation distance for x=");

        scanf("%d", &tx);

        printf("Enter the translation distance for y=");

        scanf("%d", &ty);

        rectangle(50 + tx, 50 + ty, 300 + tx, 300 + ty);

        getch();

        closegraph();

}
```

Output-



```
Enter the translation distance for x=12
Enter the translation distance for y=12
```

LAB ASSIGNMENT 10

Q10- Perform Scaling of a square whose vertex are pre-defined.

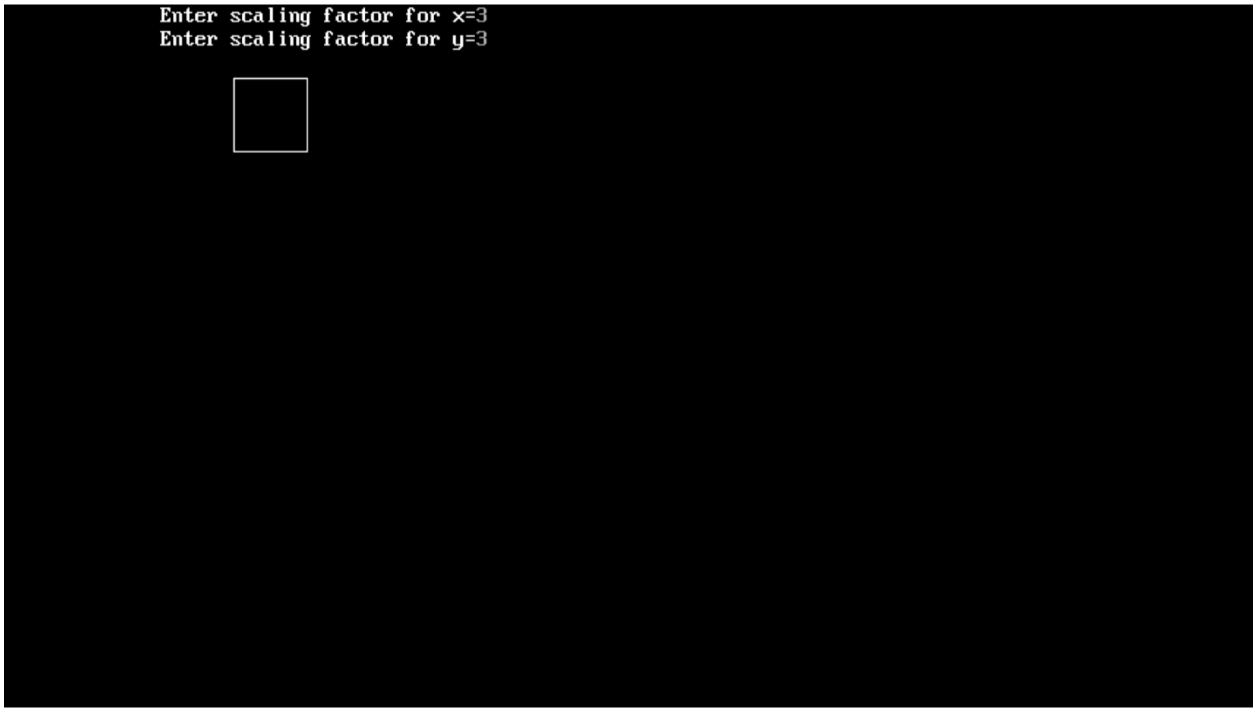
Sol-

Code-

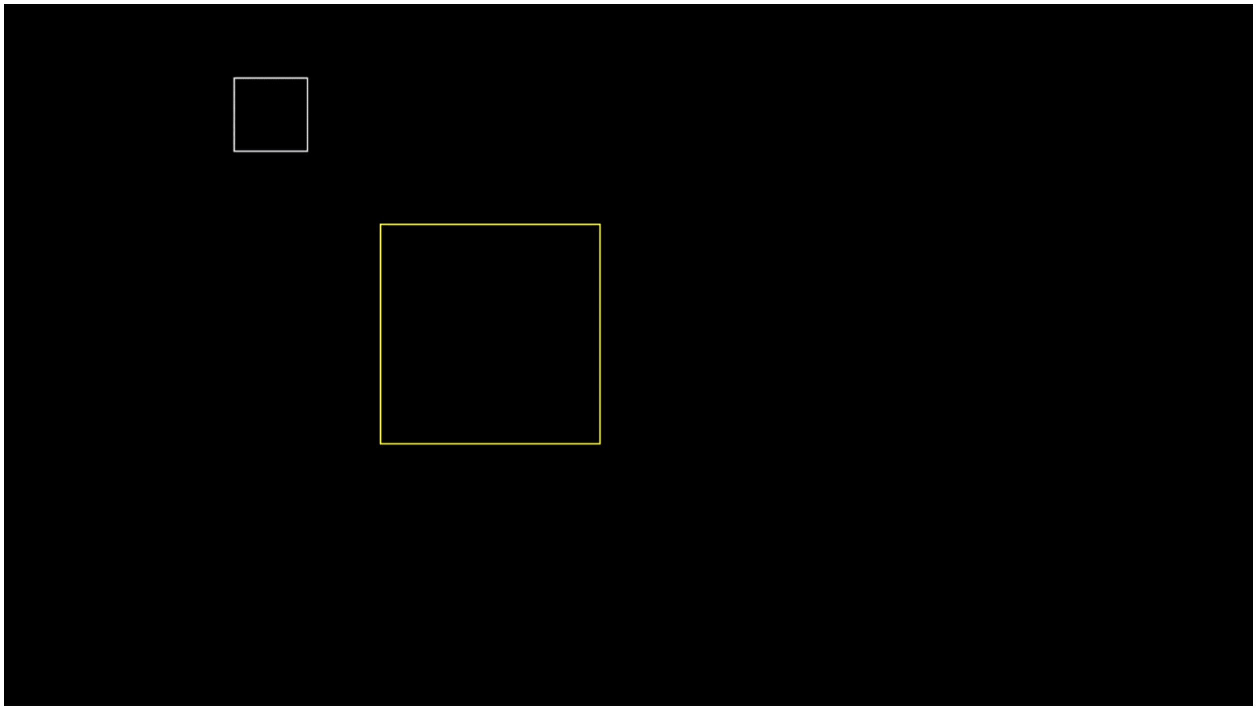
```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>


void main(){

	int gd=DETECT, gm;

	float tx, ty;

	initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

	rectangle(50, 50, 100, 100);

	printf("Enter scaling factor for x=");

	scanf("%f", &tx);

	printf("Enter scaling factor for y=");
```

```
        scanf("%f", &ty);
        clrscr();
        cleardevice();
        rectangle(50, 50, 100, 100);
        setcolor(YELLOW);
        rectangle(50*tx, 50*ty, 100*tx, 100*ty);
        getch();
        closegraph();
}
```

Output-



```
Enter scaling factor for x=3
Enter scaling factor for y=3
```



LAB ASSIGNMENT 11

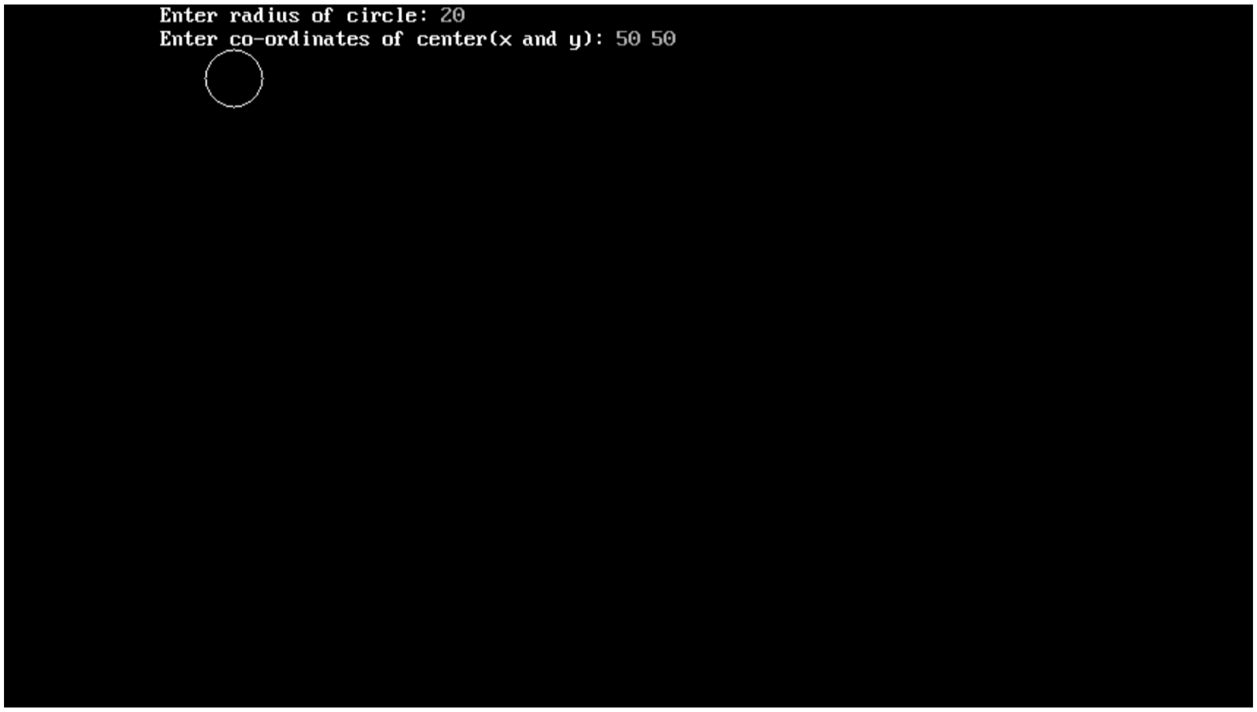
Q11- Implement code in C to draw a circle using mid point algorithm.

Sol- Code-

```
#include<stdio.h>

#include<graphics.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, WHITE);

putpixel(x0 + y, y0 + x, WHITE);

putpixel(x0 - y, y0 + x, WHITE);
putpixel(x0 - x, y0 + y, WHITE);

putpixel(x0 - x, y0 - y, WHITE);

putpixel(x0 - y, y0 - x, WHITE);

putpixel(x0 + y, y0 - x, WHITE);

putpixel(x0 + x, y0 - y, WHITE);


if (err <= 0)

{

    y += 1;

    err += 2*y + 1;

}


if (err > 0)

{

    x -= 1;

    err -= 2*x + 1;

}

    }

}


int main()

{

int gdriver=DETECT, gmode, error, x, y, r;

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


printf("Enter radius of circle: ");

scanf("%d", &r);


printf("Enter co-ordinates of center(x and y): ");
```

```
scanf("%d%d", &x, &y);

drawcircle(x, y, r);

getch();

return 0;

}
```

Output-



```
Enter radius of circle: 20
Enter co-ordinates of center(x and y): 50 50
```

LAB ASSIGNMENT 12

Q12- Implement code in C to make line reflection in x and y axis.

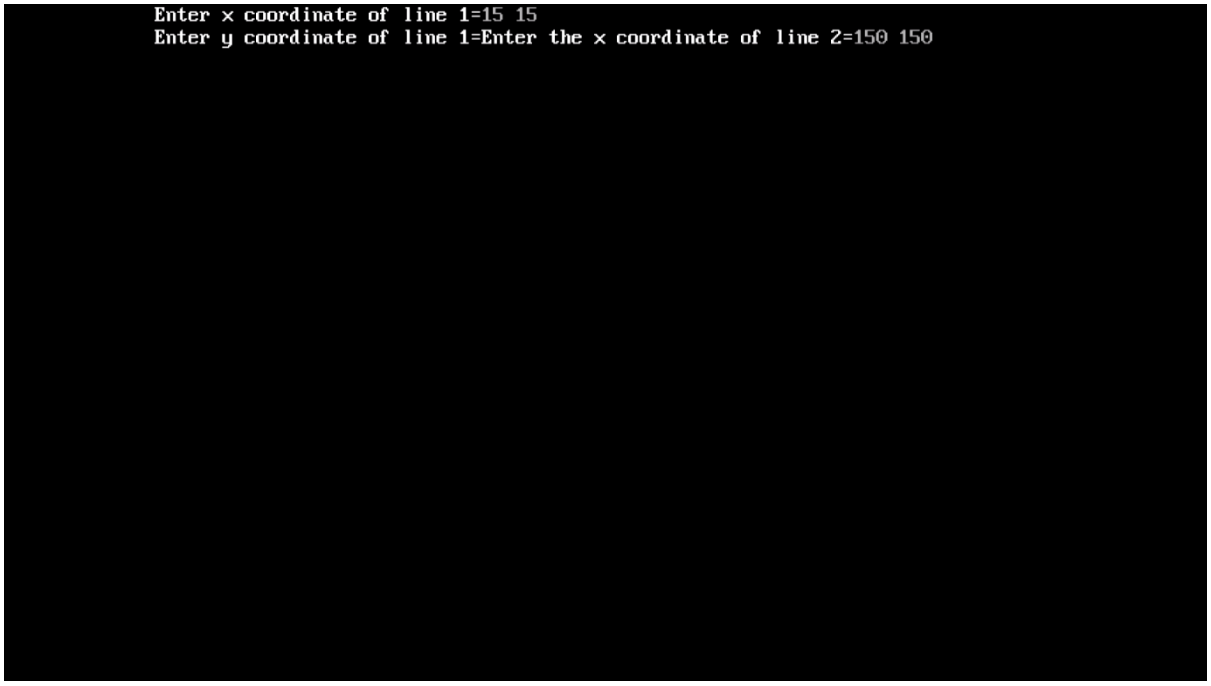
Sol-

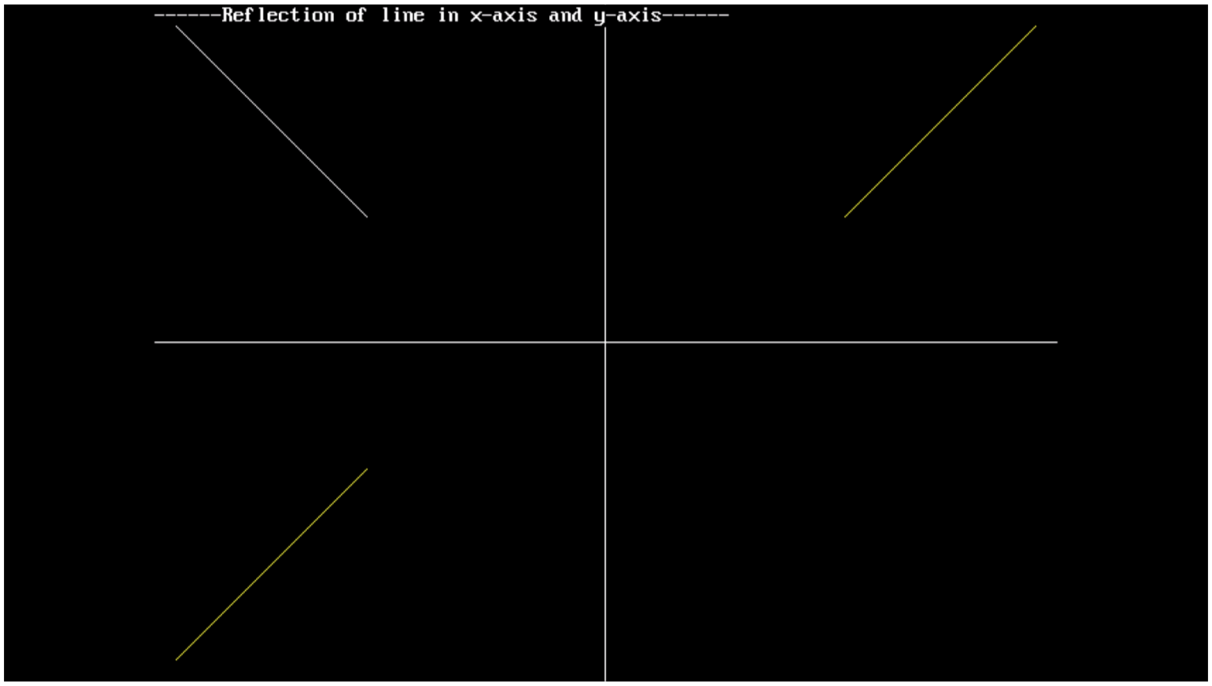
Code-

```
#include<graphics.h>

#include<stdio.h>
```

```
#include<conio.h>

void main(){
        int gd=DETECT, gm,  x1, x2, y1, y2;;
        initgraph(&gd,&gm, "c:\\turboc3\\bgi");


        printf("Enter x coordinate of line 1=");
        scanf("%d", &x1);
        printf("Enter y coordinate of line 1=");
        scanf("%d", &y1);
        printf("Enter the x coordinate of line 2=");
        scanf("%d", &x2);
        printf("Enter the y coordinate of line 2=");
        scanf("%d", &y2);
        clrscr();
        cleardevice();
        line(getmaxx()/2, 0, getmaxx()/2, getmaxy());
        line(0, getmaxy()/2, getmaxx(), getmaxy()/2);
        printf("------Reflection of line in x-axis and y-axis------");
        line(x1, y1, x2, y2);
        setcolor(YELLOW);
        line(x1, getmaxy()-y1, x2, getmaxy()-y2);
        line(getmaxx()-x1, y1, getmaxx()-x2, y2);
        getch();
        closegraph();
}
```

Output-


```
Enter x coordinate of line 1=15 15
Enter y coordinate of line 1=Enter the x coordinate of line 2=150 150
```


```
------Reflection of line in x-axis and y-axis------
```

LAB ASSIGNMENT 13

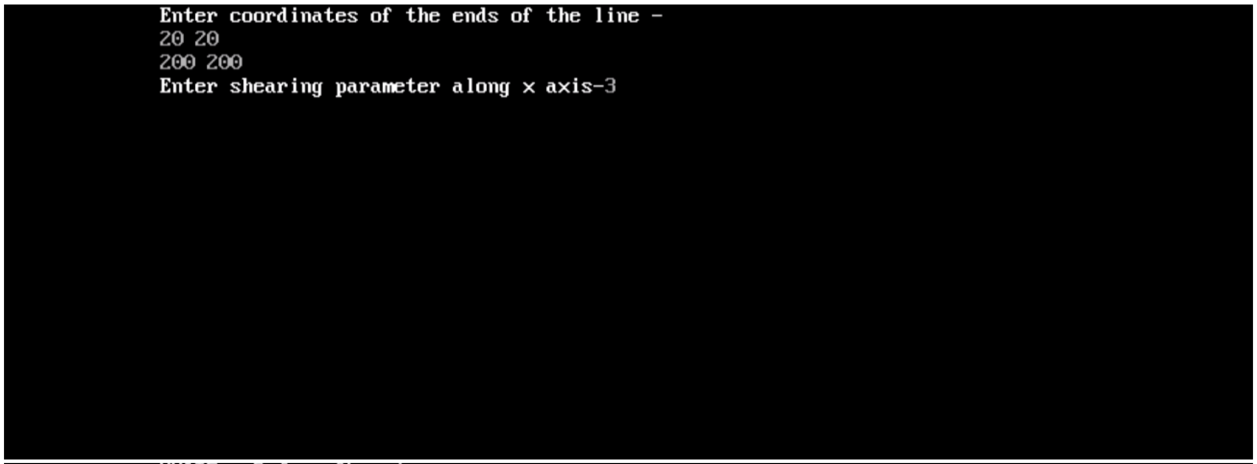
Q13- Implement a Code in C to perform line shearing along x and y axis respectively.

Sol-

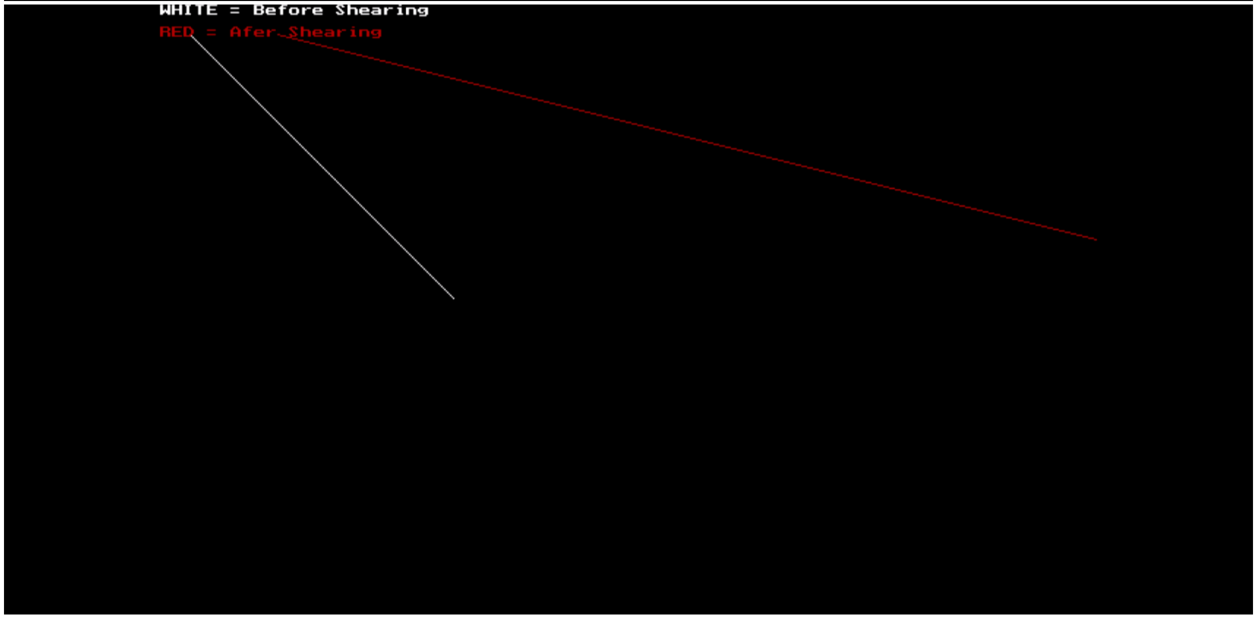
Code-

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void shearx(int x0, int y0, int x1, int y1, int shx){

        int xn0=x0+shx*y0;

        int yn0=y0;

        int xn1=x1+shx*y1;

        int yn1=y1;

        line(xn0, yn0, xn1, yn1);

}

void sheary(int x0, int y0, int x1, int y1, int shy){

        int xn0=x0;

        int yn0=y0+shy*x0;

        int xn1=x1;

        int yn1=y1+shy*x1;

        line(xn0, yn0, xn1, yn1);

}

void main(){

        int gd=DETECT, gm, x0, y0, x1, y1, shx, shy;

        initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

        printf("Enter coordinates of the ends of the line -\n");

        scanf("%d%d%d%d", &x0, &y0, &x1, &y1);

        printf("Enter shearing parameter along x axis-");

        scanf("%d",&shx);

        clrscr();

        cleardevice();

        setcolor(WHITE);
        outtextxy(0, 0, "WHITE = Before Shearing");

        line(x0, y0, x1, y1);

        setcolor(RED);

        outtextxy(0, 15, "RED = Afer Shearing");

        shearx(x0, y0, x1, y1, shx);

        getch();

        cleardevice();

        printf("Enter shearing parameter along y axis-");

        scanf("%d", &shy);

        cleardevice();

        setcolor(WHITE);

        outtextxy(0, 0, "WHITE = Before Shearing");

        line(x0, y0, x1, y1);

        setcolor(RED);

        outtextxy(0, 15, "RED = Afer Shearing");

        sheary(x0, y0, x1, y1, shx);

        getch();

        closegraph();


}
```

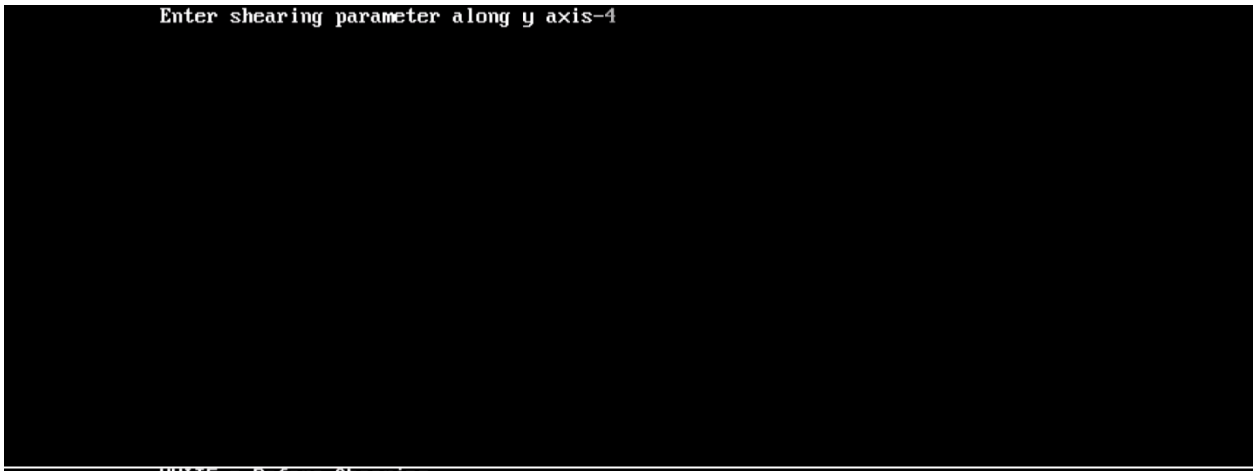
Output-



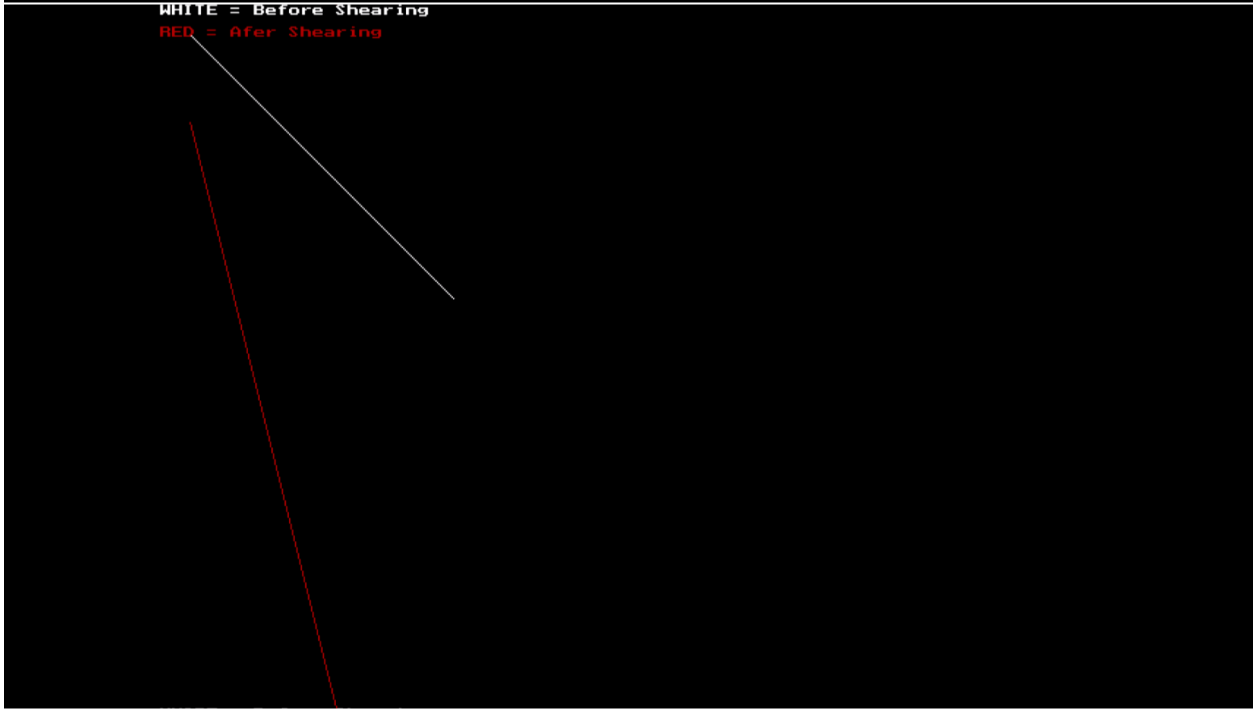
```
Enter coordinates of the ends of the line -
20 20
200 200
Enter shearing parameter along x axis-3
```



```
WHITE = Before Shearing
RED = Afer Shearing
```



```
Enter shearing parameter along y axis-4
```



```
WHITE = Before Shearing
RED = Afer Shearing
```

LAB ASSIGNMENT 14

Q14- Write a C program to perform line clipping using Cohen Sutherland Algorithm.

Sol-

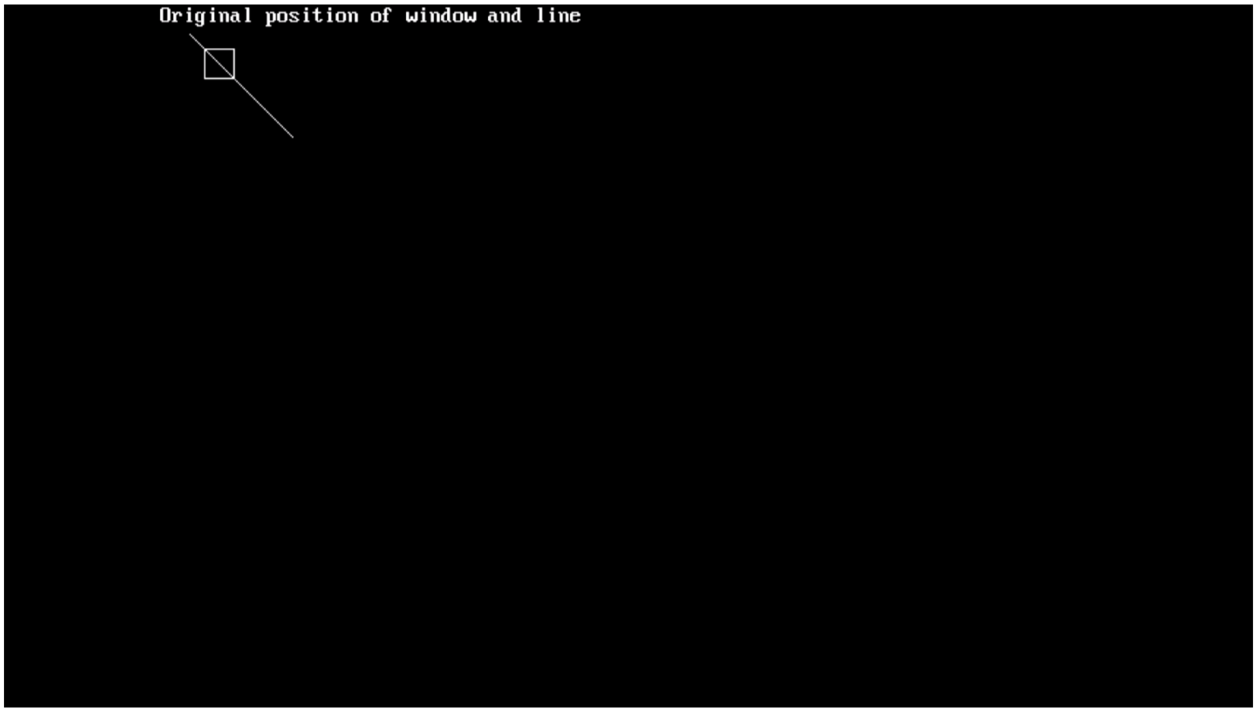
Code-

```
#include <stdio.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) {
```

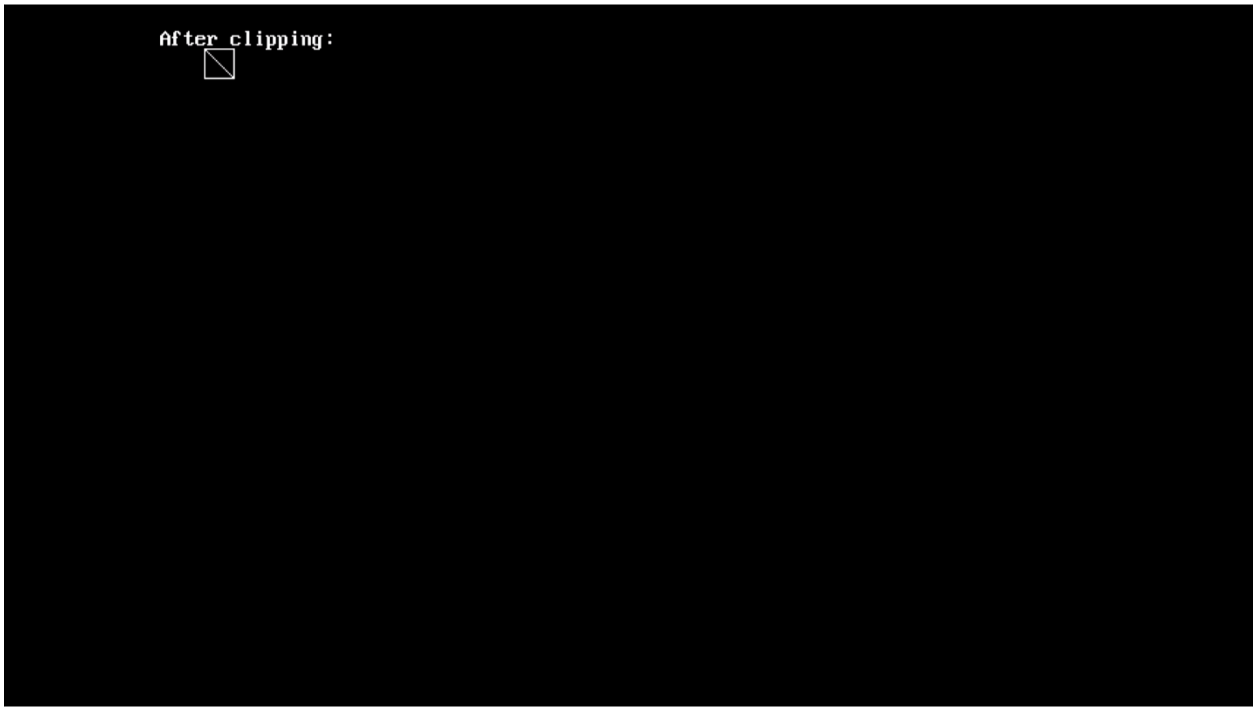
```
    accept = TRUE;
    done = TRUE;
} else if (outcode1 & outcode2) {
    done = TRUE;
} else {
    double x, y;
    int outcode_ex = outcode1 ? outcode1 : outcode2;
    if (outcode_ex & TOP) {
            x = x1 + (x2 - x1) * (ymax - y1) / (y2 - y1);
            y = ymax;
    } else if (outcode_ex & 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 = DETECT;

    printf("Enter the number of lines to be clipped-");
    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-bottom corners of the clip window:\n");
    for (i = 0; i < 4; i++) scanf("%d", & clip[i]);

    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
    printf("Original position of window and line\n");
    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();
    printf("After clipping:\n");
    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-



```
Original position of window and line
```



```
After clipping:
```

LAB ASSIGNMENT 15

Q15- Write a C program to perform line clipping using Liang Barsky Algorithm.

Sol-

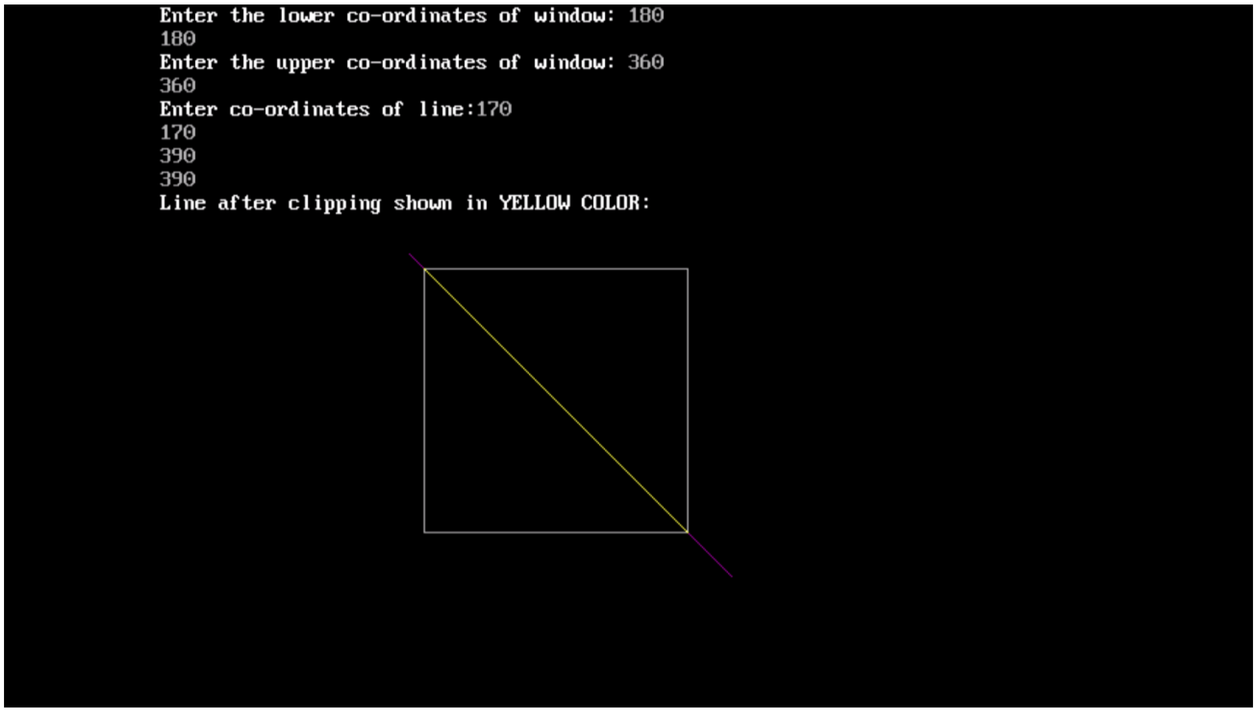
Code-

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

        initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
        printf("Enter the lower co-ordinates of window: ");
```

```
scanf("%d%d",&xmin,&ymin);

printf("Enter the upper co-ordinates of window: ");

scanf("%d%d",&xmax,&ymax);

setcolor(5);

rectangle(xmin,ymin,xmax,ymax);

printf("Enter co-ordinates of line:");

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

cleardevice();

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)

                        printf("line completely outside the window");

                else

                        if(p[i]==0 && q[i] >= 0)

                                printf("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){
		setcolor(7);
		xx1=x1+t1*dx;
		xx2=x1+t2*dx;
		yy1=y1+t1*dy;
		yy2=y1+t2*dy;
		printf("Line after clipping shown in YELLOW COLOR:");
		rectangle(xmin,ymin,xmax,ymax);
		setcolor(YELLOW);
		line(xx1,yy1,xx2,yy2);
	}
	else{
		printf("Line lies out of the window");
	}
	getch();
}
```

Output-



```
Enter the lower co-ordinates of window: 180
180
Enter the upper co-ordinates of window: 360
360
Enter co-ordinates of line:170
170
390
390
Line after clipping shown in YELLOW COLOR:
```