1. Write C program for DDA line generation if user can input endpoints of line

Source code:

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

int main(){
    int x, y, x1, y1, x2, y2, dx, dy, step, xinc, yinc, i, gd = 0, gm;
    initgraph(&gd, &gm, "");
    setbkcolor(WHITE);
    printf("Enter the start points: ");
    scanf("%d%d", &x1, &y1);
    printf("Enter the endpoints: ");
    scanf("%d%d", &x2, &y2);
    dx = (x2 - x1);
    dy = (y2 - y1);
    if(abs(dx) >= abs(dy))
            step = abs(dx);
    else
        step = abs(dy);
     xinc = dx / step;
    yinc = dy / step;
    x = x1;
    y = y1;
    putpixel(x,y,4);
    for(i = 0; i < step; i++){
        x = x + xinc;
        y = y + yinc;
        putpixel(x,y,WHITE);
    }
    getch();
    closegraph();
    return 0;
}
```

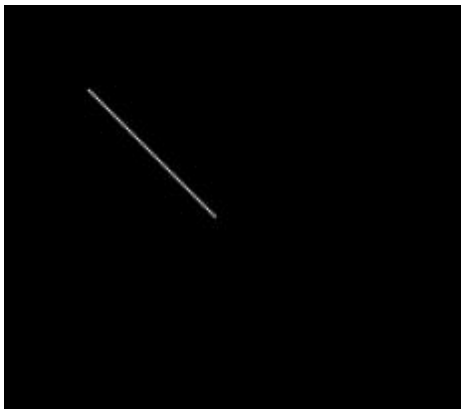
OUTPUT:

2. Write C program for DDA Line generation of End points A(10,12) and B(15,17)

Source code:

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

int main(){
   int x, y, x1 = 100, y1 = 120, x2 = 150, y2 = 170, dx, dy, step, xinc, yinc, i, gd = 0,
gm;
   initgraph(&gd, &gm, "");
   setbkcolor(WHITE);
   dx = (x2 - x1);
   dy = (y2 - y1);
   if(abs(dx) >= abs(dy))
         step = abs(dx);
   else
      step = abs(dy);
   xinc = dx / step;
   yinc = dy / step;
   x = x1;
   y = y1;
   putpixel(x,y,4);
   for(i = 0; i < step; i++){
      x = x + xinc;
      y = y + yinc;
      putpixel(x,y,4);
   }
   getch();
   closegraph();
   return 0;
}
```

OUTPUT:

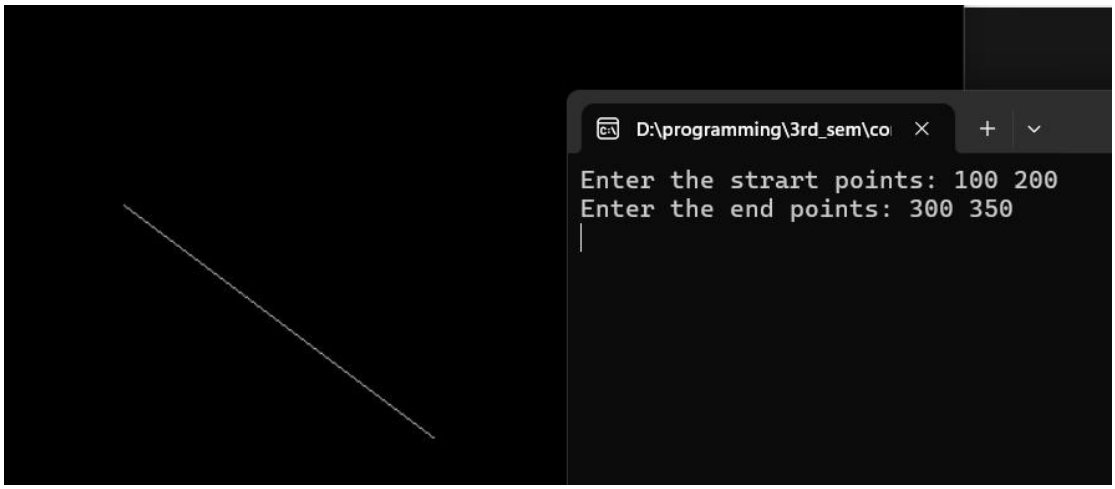


3. Write C program for BSA line generation if user can input endpoints of line

Source code:

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

void bsa(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;
    }
    getch();
}

int main(){
    int gd = 0, gm, x0, y0, x1, y1;
    initgraph(&gd, &gm, "");
    printf("Enter the strart points: ");
    scanf("%d%d", &x0, &y0);
    printf("Enter the end points: ");
    scanf("%d%d", &x1, &y1);
    bsa(x0, y0, x1, y1);
    closegraph();
}
```

OUTPUT:

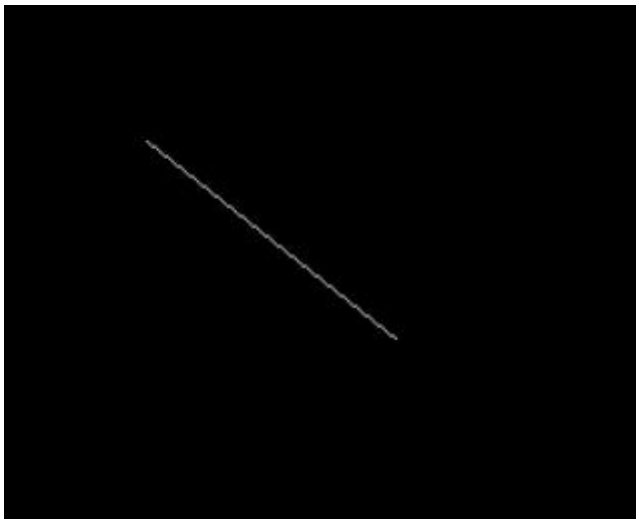


4. Write c Program For Bsa Line Generation Of End Points a(80,120) And b(180,200)

Source Code:

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

void bsa(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;
    }
    getch();
}

int main(){
    int gd = 0, gm, x0 = 80, y0 = 120, x1 = 180, y1 = 200;
    initgraph(&gd, &gm, "");
    setbkcolor(red);
    bsa(x0, y0, x1, y1);
    closegraph();
}
```

OUTPUT:

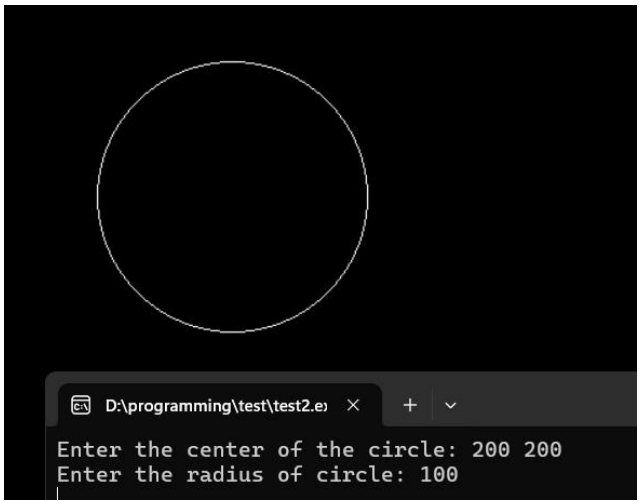


5. Write a c program for Generating All points of Circle by mid-point Algorithm

Source Code:

```
#include<stdio.h>
#include<graphics.h>
int main(){
    int xc, yc, x, y, r, gd = 0, gm, p, i;
    initgraph(&gd, &gm, "");
    printf("Enter the center of the circle: \n");
    scanf("%d%d", &xc, &yc);
    printf("Enter the radius of circle: \n");
    scanf("%d", &r);
    x = 0, y = r, p = 1 - r;
    do {
        putpixel( xc + x, yc + y, RED);
        putpixel( xc + x, yc - y, RED);
        putpixel( xc - x, yc + y, RED);
        putpixel( xc - x, yc - y, RED);
        putpixel( xc + y, yc + x, RED);
        putpixel( xc + y, yc - x, RED);
        putpixel( xc - y, yc + x, RED);
        putpixel( xc - y, yc - x, RED);
        if(p < 0)  {
            x = x + 1;
            y = y;
            p = p + 2 * x + 2;
        }
        else{
            x = x + 1;
            y = y - 1;
            p = p + 2 * (x - y) + 1;
        }
    } while (x < y);
    getch();
    closegraph();
}
```

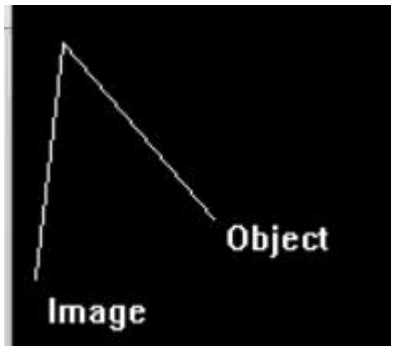
OUTPUT:



6. Rotate line AB whose endpoints are A (20, 50) and B (80, 120) about origin through a 30° clockwise direction.

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#define PI 3.14159265

int main()
{
   int gd = DETECT, gm, x1, y1, x2, y2, xn, yn;
   double r11, r12, r21, r22, th;
   x1 = 20, y1 = 50, x2 = 80,  y2 = 120 ,th = 30;

   initgraph(&gd, &gm,"");
   line(x1, y1, x2, y2);
   outtextxy(x2 + 5, y2, "Object");
   th = th * PI / 180.0;
   r11 = cos(th);
   r12 = sin(th);
   xn = round(x2 * r11 - y2 * r12);
   yn = round(x2 * r12 + y2 * r11);
   line(x1, y1, xn, yn);
   outtextxy(xn + 5, yn + 5, "Image");
   getch();
   closegraph();
   return 0;
}
```

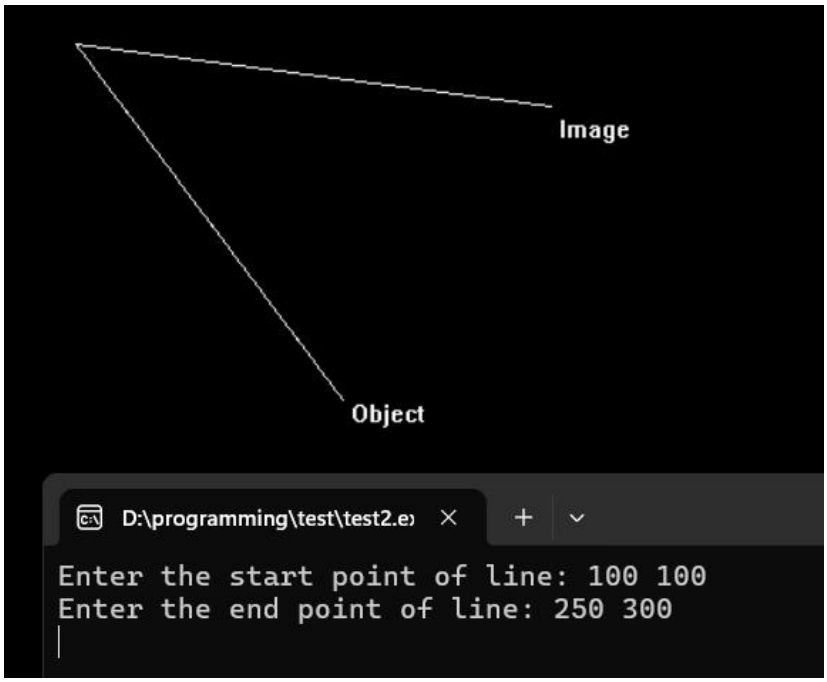
OUTPUT:



7. Rotate line AB whose end points input by user about origin through a 30° anti clockwise direction

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#define PI 3.14159265

int main()
{
    int gd = DETECT, gm, x1, y1, x2, y2, xn, yn;
    double r11, r12, r21, r22, th;
    printf("Enter the start point of line: ");
    scanf("%d%d", &x1, &y1);
    printf("Enter the end point of line: ");
    scanf("%d%d", &x2, &y2);
    th = 30.0;
    initgraph(&gd, &gm,"");
    line(x1, y1, x2, y2);
    outtextxy(x2 + 5, y2, "Object");
    th = th * PI / 180.0;
    r11 = cos(th);
    r12 = -sin(th);
    xn = round(x2 * r11 - y2 * r12);
    yn = round(x2 * r12 + y2 * r11);
    line(x1, y1, xn, yn);
    outtextxy(xn + 5, yn + 5, "Image");

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

OUTPUT:

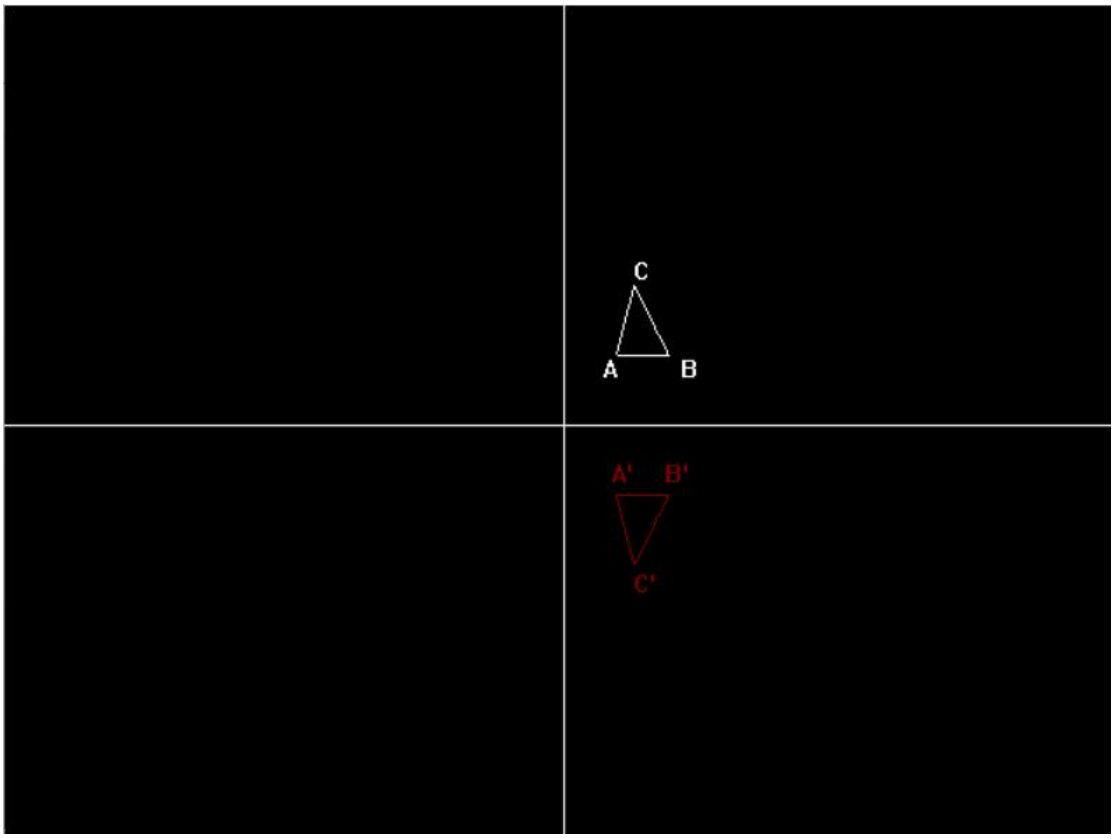


8. A triangle ABC is given. The coordinates of A, B, C are given as A (3 , 4), B (6 , 4), C (4 , 8) Write a program to reflected position of triangle i.e., to the x-axis.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
# define absx(x) x+320
# define absy(y) 240-y

void DrawCordinates()
{
  line(320,0,320,640);
  line(0,240,640,240);
}

int main()
{
        int gd=DETECT, gm;
        int k=0,x1 ,y1;
        int poly[20], i;
        initgraph(&gd,&gm,"");
        DrawCordinates();
        poly[0]=absx(30);
        poly[1]=absy(40);
        poly[2]=absx(60);
        poly[3]=absy(40);
        poly[4]=absx(40);
        poly[5]=absy(80);
        poly[6]=poly[0];
        poly[7]=poly[1];
        drawpoly(4,poly);
        outtextxy(poly[0] - 7, poly[1], "A");
        outtextxy(poly[2] + 7, poly[3], "B");
        outtextxy(poly[4], poly[5] - 16, "C");
                printf("\nX-axis\n");

                for(k=0;k<8;k=k+2)
                {
                        x1=poly[k];
                        y1=poly[k+1];
                        poly[k]=x1;
                        poly[k+1]=240-(y1-240);
                }
                poly[8]=poly[0];
                poly[9]=poly[1];
                setcolor(RED);
                drawpoly(4,poly);
                outtextxy(poly[0] - 2, poly[1] - 20, "A'");
                outtextxy(poly[2] - 2, poly[3] - 20, "B'");
                outtextxy(poly[4], poly[5] + 3, "C'");
```

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

OUTPUT:

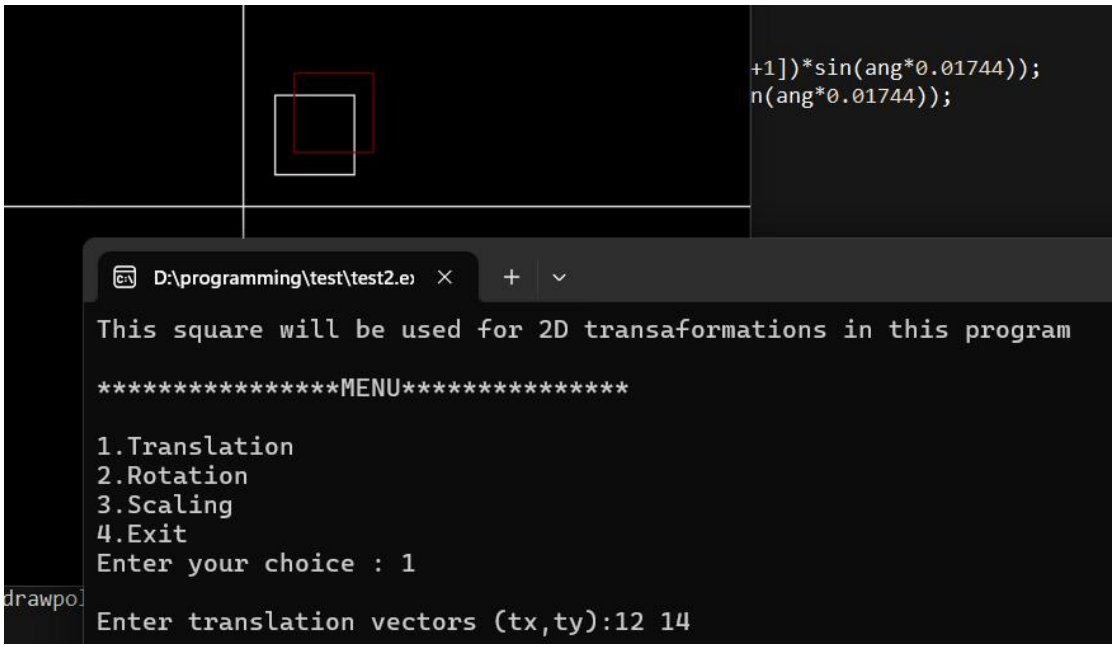


9. Write a program to perform following. Transformation, rotation, scaling

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
# define absx(x) x+320
# define absy(y) 240-y
void DrawCordinates(){
  line(320,0,320,640);
  line(0,240,640,240);
}
int main(){
        int gd=DETECT,gm;
        int ang,sx,sy,c,k=0,x1,y1,sh,shx,shy,c1;
        int poly[20],choice,i,tdx,tdy,shchoice, x,y;
        initgraph(&gd,&gm,"");
        DrawCordinates();
        poly[0]=absx(20);
        poly[1]=absy(20);
        poly[2]=absx(70);
        poly[3]=absy(20);
        poly[4]=absx(70);
        poly[5]=absy(70);
        poly[6]=absx(20);
        poly[7]=absy(70);
        poly[8]=poly[0];
        poly[9]=poly[1];
        drawpoly(5,poly);
        printf("This square will be used for 2D transaformations in this program\n");
                printf("\n***************MENU**************\n");
                printf("\n1.Translation\n2.Rotation\n3.Scaling\n4.Exit");
        do{
                printf("\nEnter your choice : ");
                scanf("%d",&choice);
                switch(choice){
                        case 1:
                                printf("\nEnter translation vectors (tx,ty):");
                                scanf("%d%d",&tdx,&tdy);
                                for(i=0;i<8;i=i+2){
                                        poly[i]=poly[i]+tdx;
                                        poly[i+1]-=tdy;
                                }
                                poly[8]=poly[0];
                                poly[9]=poly[1];
                                setcolor(RED);
                                drawpoly(5,poly);
                                break;
                        case 2:
                                printf("\nEnter rotation  angle:");
                                scanf("%d",&ang);
                                for(i=0;i<8;i=i+2){
```

```
                                        int tx= poly[i]-320;
                                        poly[i]=320+((poly[i]-320)*cos(ang*0.01744)-
(240-poly[i+1])*sin(ang*0.01744));
                                        poly[i+1]=240-((240-
poly[i+1])*cos(ang*0.01744)+(tx)*sin(ang*0.01744));
                                }
                                poly[8]=poly[0];
                                poly[9]=poly[1];
                                setcolor(RED);
                                drawpoly(5,poly);
                                break;
                        case 3:
                                printf("\nEnter scaling vectors (tx,ty):");
                                scanf("%d%d",&sx,&sy);
                                for(i=2;i<8;i=i+2){
                                        poly[i]=poly[i]*sx+poly[0]*(1-sx);
                                        poly[i+1]=poly[i+1]*sy+poly[1]*(1-sy);
                                }
                                poly[8]=poly[0];
                                poly[9]=poly[1];
                                setcolor(RED);
                                drawpoly(5,poly);
                                break;
                        case 4:
                                exit(0);
                        }
        }while(choice!=4);
                getch();
        closegraph();
}
```

OUTPUT:

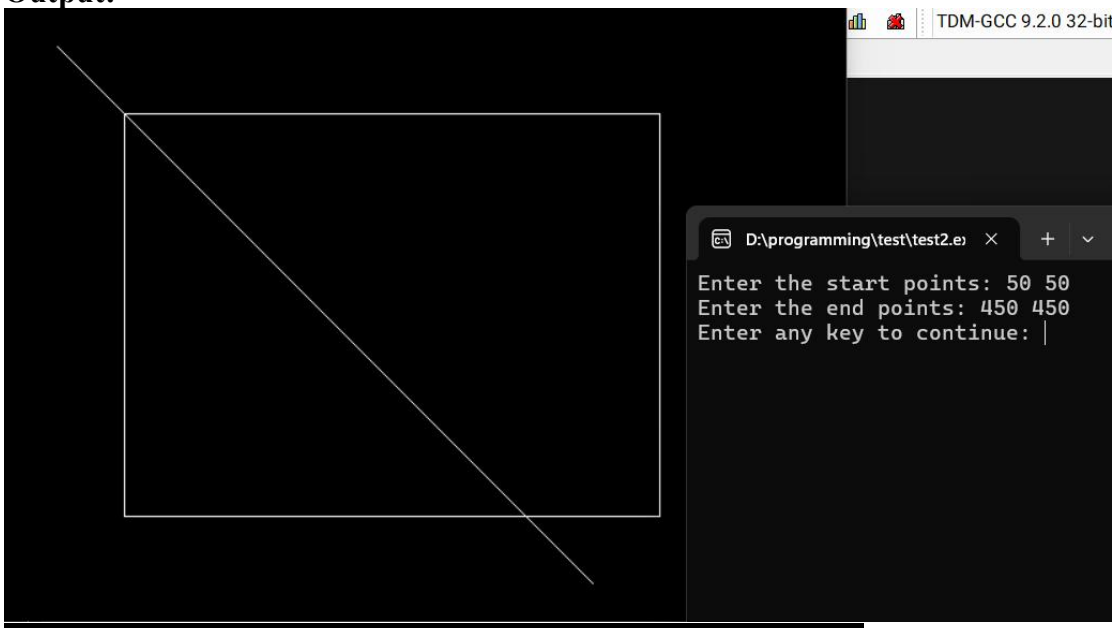


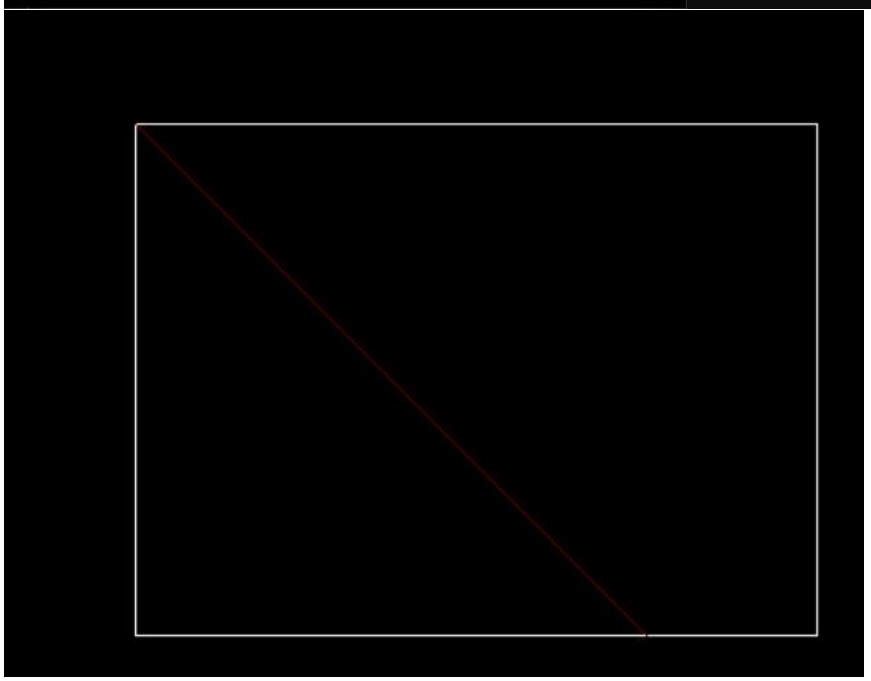
10. Write a program to perform Line Clipping using Cohen Sutherland Algorithm user can input coordinates of line.

```
#include<stdio.h>
#include<graphics.h>
static int xmin, ymin, xmax, ymax, LEFT = 1, RIGHT = 2, BOTTOM = 4, TOP =
        8;
int getcode(int x, int y){
        int code = 0;
        if(y>ymax) code |= TOP;
        if(y<ymin) code |= BOTTOM;
        if(x>xmax) code |= LEFT;
        if(x<xmin) code |=RIGHT;
        return code;
}

int main(){
        int gd=0, gm, x1, x2, y1, y2;
        int outcode1, outcode2, accept = 0, x, y, temp;
        float m;
        char a, b;
        initgraph(&gd, &gm,"");
        setcolor(WHITE);
        xmin = ymin = 100, xmax = 500, ymax = 400;
        rectangle(xmin,ymin, xmax, ymax)
        printf("Enter the start points: ");
    scanf("%d%d", &x1, &y1);
    printf("Enter the end points: ");
    scanf("%d%d", &x2, &y2);
        line(x1, y1, x2, y2);
        outcode1 = getcode(x1,y1);
        outcode2 = getcode(x2,y2);
        printf("Enter any key to continue: ");
    scanf("%c%c", &a,&b);
    closegraph();
    initgraph(&gd, &gm, "");
        while(1){
                m = (float)(y2 - y1) / (x2 - x1);
                if(outcode1 == 0 && outcode2 == 0){
                        accept = 1;
                        break;
                }
                else if((outcode1 & outcode2) != 0)
                        break;
                else{
                        if(outcode1 == 0)
                                temp = outcode2;
                        else
                                temp = outcode1;
                        if(temp & TOP){
                                x = x1 + (ymax - y1) / m;
```

```
                        y = ymax;
                }
                else if(temp & BOTTOM){
                        x = x1 + (ymin - y1) / m;
                        y = ymin;
                }
                else if(temp & LEFT){
                        x = xmin;
                        y = y1 + m * (xmin - x1);
                }
                else if(temp & RIGHT){
                        x = xmax;
                        y = y1 + m * (xmax - x1);
                }
                if(temp == outcode1){
                        x1 = x;
                        y1 = y;
                        outcode1 = getcode(x1, y1);
                }
                else{
                        x2 = x;
                        y2 = y;
                        outcode2 = getcode(x2, y2);
                }
        }
    }
    printf("\nAfter clipping: ");
    if(accept){
        system("cls");
        rectangle(xmin,ymin, xmax, ymax);
        setcolor(RED);
        line(x1, y1, x2, y2);
    }
    getch();
    closegraph();
}
```

Output:


TDM-GCC 9.2.0 32-bit
D:\programming\test\test2.e
Enter the start points: 50 50
Enter the end points: 450 450
Enter any key to continue:

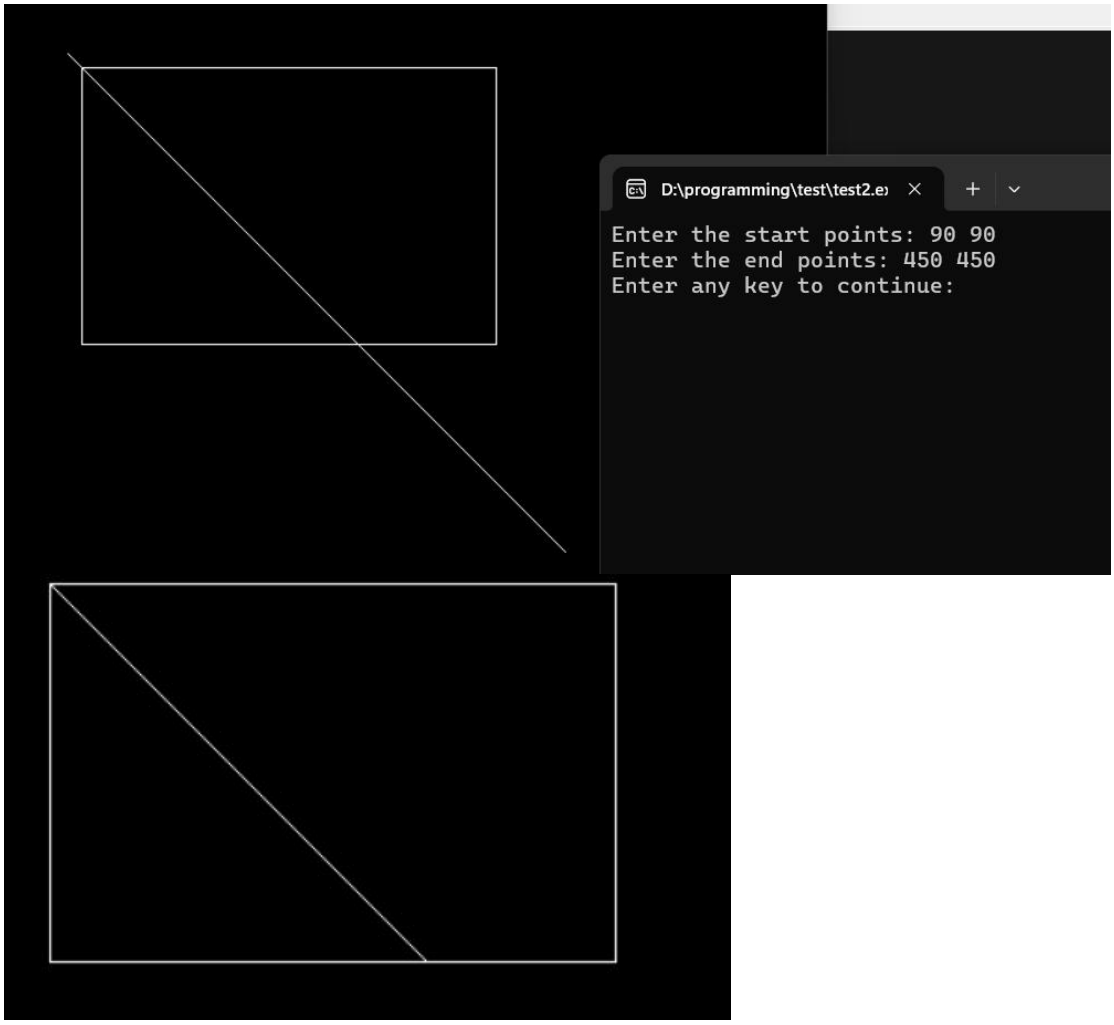


11. Program to clip a line using Liang Barsky Metho

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
int main() {
    int i, gd = DETECT, gm;
    int x1, y1, x2, y2, xmin, ymin, xmax, ymax, xx1, xx2, yy1, yy2, dx, dy;
    float t1, t2, p[4], q[4], temp;
    char a, b;
    xmin = ymin = 100, xmax = 400, ymax = 300;
    printf("Enter the start points: ");
    scanf("%d%d", &x1, &y1);
    printf("Enter the end points: ");
    scanf("%d%d", &x2, &y2);
    initgraph(&gd, &gm, "");
    rectangle(xmin, ymin, xmax, ymax);
    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;

    printf("Enter any key to continue: ");
    scanf("%c%c", &a,&b);
    closegraph();

    initgraph(&gd, &gm, "");
    rectangle(xmin, ymin, xmax, ymax);
    for(i = 0; i < 4; i++)
    {
        if(p[i] == 0) {
            printf("The line is parallel to one of the clipping boundary.\n");
            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);
            }
        }
        getch();
        closegraph();
    }
}

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);
}
getch();
closegraph();
return 0;
}
```

OUTPUT:


D:\programming\test\test2.e
Enter the start points: 90 90
Enter the end points: 450 450
Enter any key to continue: