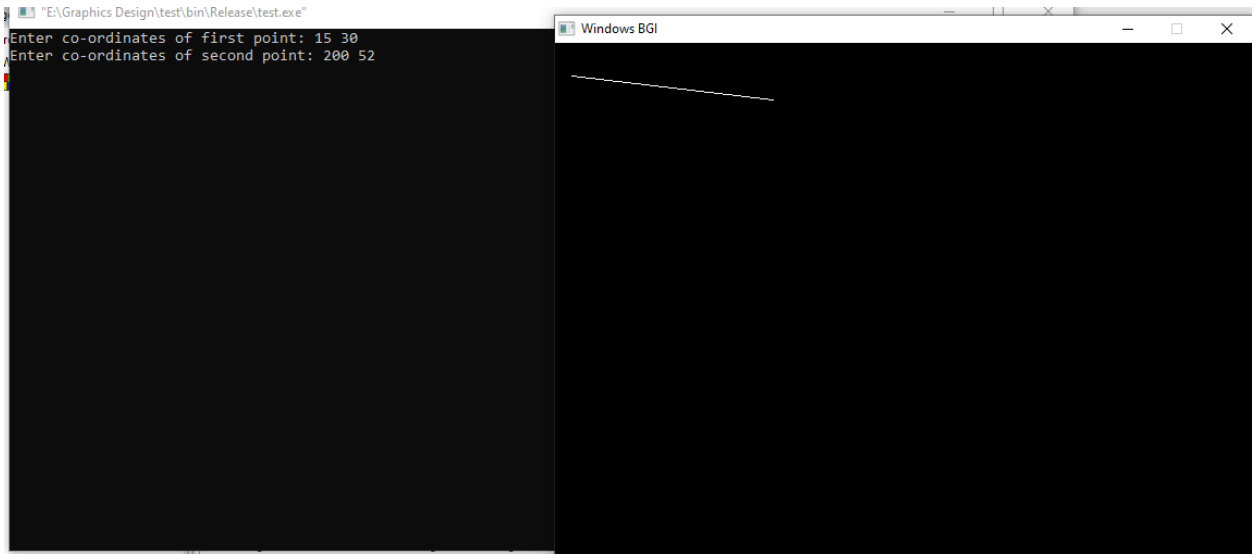
Experiment 1: Bresenham's Line Drawing

Code:

```
#include<bits/stdc++.h>
#include <graphics.h>
using namespace std;

void bresenhamsLine(int x0, int y0, int x1, int y1){
    if(x1<x0){
        swap(x0,x1);
        swap(y0,y1);
    }
    int dx = abs(x1-x0);
    int dy = abs(y1-y0);
    int p = 2*dy - dx;
    int x,y=y0;
    for(x=x0; x<=x1; x++){
        putpixel(x,y,15);
        if(p<0) p += 2*dy;
        else{
            y++;
            p += 2*(dy-dx);
        }
    }
}

int main() {
    int gd=DETECT, gm=DETECT;
    initgraph(&gd,&gm,"");

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

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

    bresenhamsLine(x0, y0, x1, y1);
    getch();
    closegraph();
    return 0;
}
```

Output:


"E:\Graphics Design\test\bin\Release\test.exe"
Enter co-ordinates of first point: 15 30
Enter co-ordinates of second point: 200 52
Windows BGI

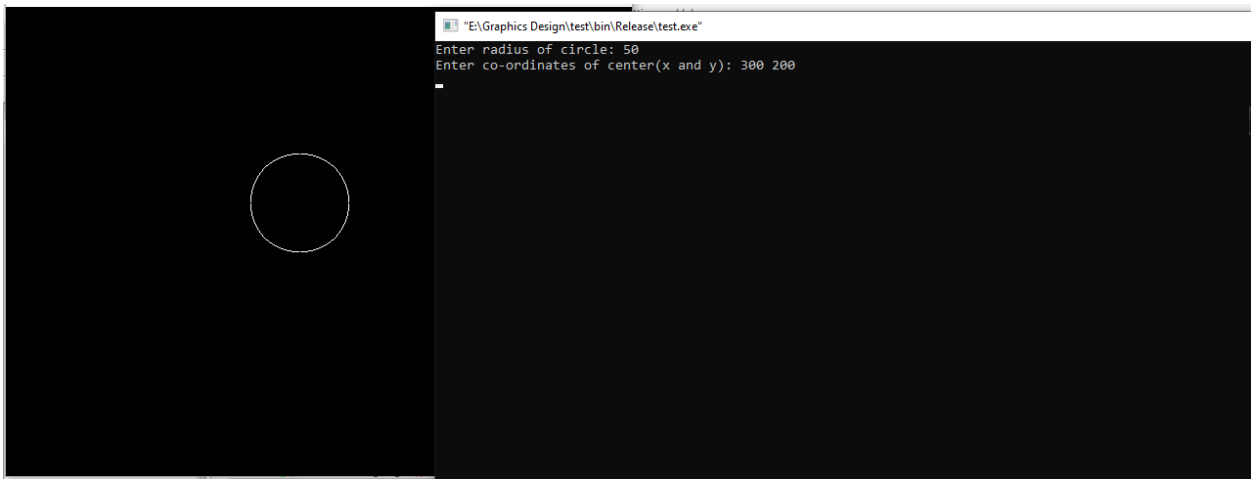
Experiment 2: Midpoint Circle Drawing

Code:

```
#include <graphics.h>

void plotpixel(int g,int h,int x,int y){
    putpixel(g+x,h+y,15);
    putpixel(g-x,h+y,15);
    putpixel(g+x,h-y,15);
    putpixel(g-x,h-y,15);
    putpixel(g+y,h+x,15);
    putpixel(g-y,h+x,15);
    putpixel(g+y,h-x,15);
    putpixel(g-y,h-x,15);
}

void midpointCircle(int g,int h,int r){
    int p,x,y;
    p = 1-r;
    x = 0;
    y = r;
    while(x<y){
        x++;
        plotpixel(g,h,x,y);
        if(p<0){
            p += 2*x + 1;
        }
        else{
            y--;
            p += 2*x + 1 - 2*y;
        }
        delay(1);
    }
}

int main() {
    int gd=DETECT, gm=DETECT;
    initgraph(&gd,&gm,"");

    int r, x0, y0;
    printf("Enter radius of circle: ");
    scanf("%d", &r);
```

```
    printf("Enter co-ordinates of center(x and y): ");
    scanf("%d%d", &x0, &y0);

    midpointCircle(x0, y0, r);

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

Output:

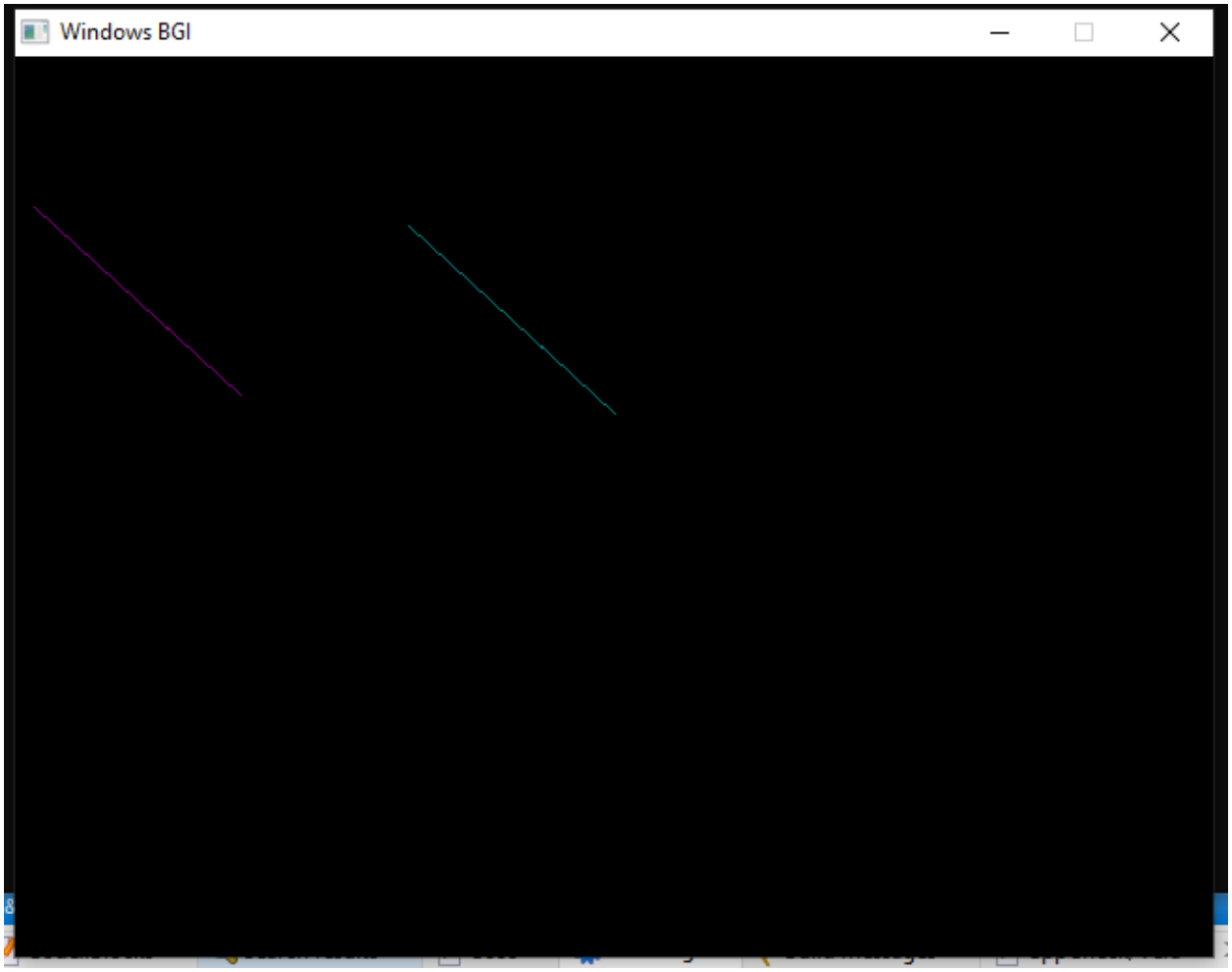


Experiment 3: 2D Translation, Rotation and Scaling

2D Translation

```
#include<stdio.h>
#include<graphics.h>
void translateLine (int P[][2], int T[]){
   setcolor (5);
   line(P[0][0], P[0][1], P[1][0], P[1][1]);
   P[0][0] = P[0][0] + T[0];
   P[0][1] = P[0][1] + T[1];
   P[1][0] = P[1][0] + T[0];
   P[1][1] = P[1][1] + T[1];
   setcolor(3);
   line(P[0][0], P[0][1], P[1][0], P[1][1]);
   getch();
   closegraph();
}

int main(){
   int gd = DETECT, gm, errorcode;
   initgraph (&gd, &gm, NULL);
   int P[2][2] = {10, 80, 120, 180},T[] = {200, 10};
   translateLine (P, T);
   return 0;
}
```

Output:

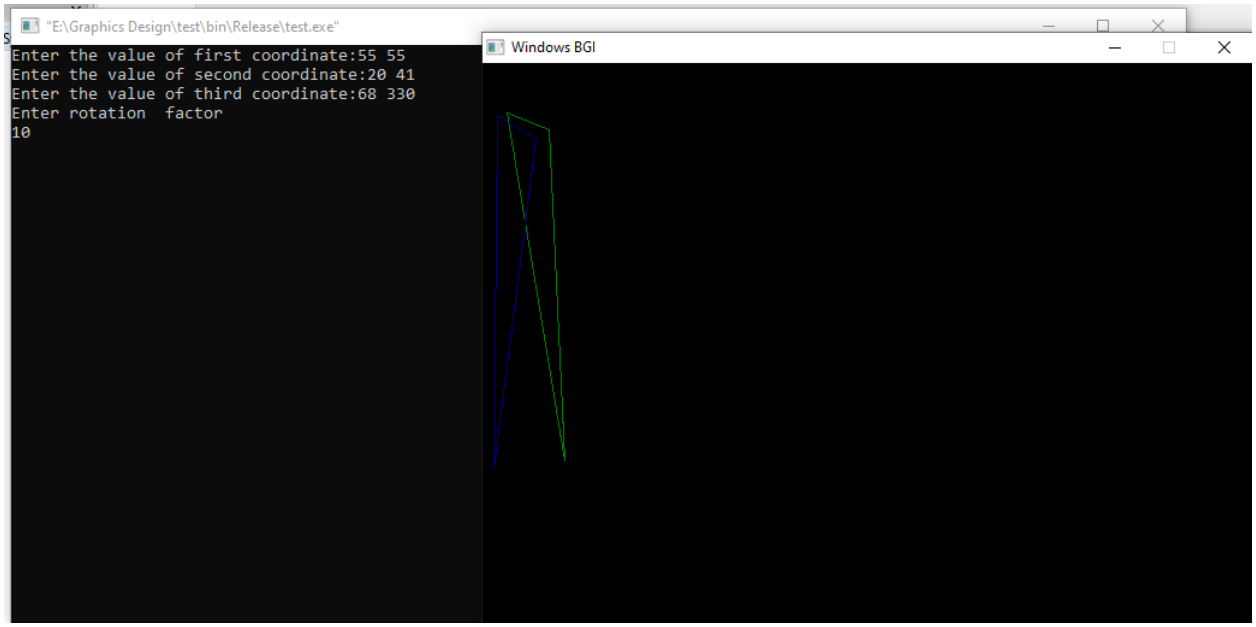


2D Rotation(triangle):

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

float r,t;

void linerotate (float a, float b, float c, float d){

      line
(a*cos(t)-b*sin(t),a*sin(t)+b*cos(t),c*cos(t)-d*sin(t),c*sin(t)+d*cos(t));
}

int main(){
   int gd = DETECT, gm;
```

```
    initgraph(&gd, &gm, "C://TurboC3//BGI");
    int X,Y,X1,Y1,X2,Y2;
    printf("Enter the value of first coordinate:");
    scanf("%d %d",&X,&Y);
    printf("Enter the value of second coordinate:");
    scanf("%d %d",&X1,&Y1);
    printf("Enter the value of third coordinate:");
    scanf("%d %d",&X2,&Y2);
    printf("Enter rotation  factor\n");
    scanf("%f",&r);
    t = (3.14/180)*r;

    setcolor(GREEN);
    line(X,Y,X1,Y1);
    line(X1,Y1,X2,Y2);
    line(X2,Y2,X,Y);

    setcolor(BLUE);
    linerotate(X,Y,X1,Y1);
    linerotate(X1,Y1,X2,Y2);
    linerotate(X2,Y2,X,Y);
    getch();
    closegraph();
}
```

Output:

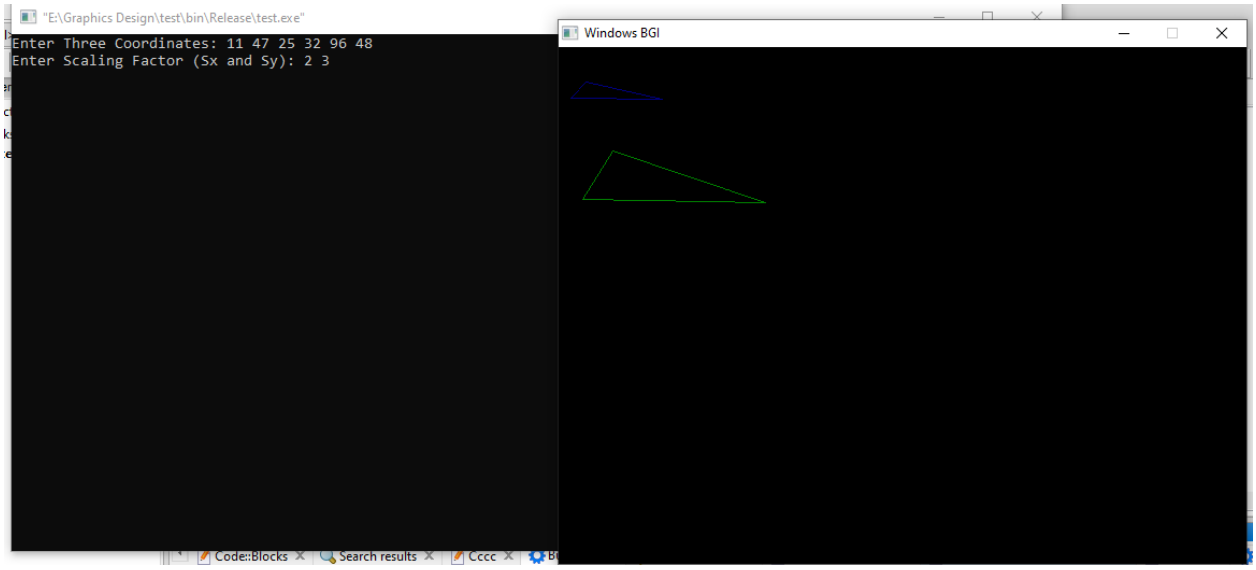


2D Scaling(Triangle):

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

void findNewCoordinate(int s[][2], int p[][1])
{
   int temp[2][1] = { 0 };

   for (int i = 0; i < 2; i++)
       for (int j = 0; j < 1; j++)
           for (int k = 0; k < 2; k++)
               temp[i][j] += (s[i][k] * p[k][j]);

   p[0][0] = temp[0][0];
   p[1][0] = temp[1][0];
}


void scale(int x[], int y[], int sx, int sy)
{
        setcolor(BLUE);
   line(x[0], y[0], x[1], y[1]);
   line(x[1], y[1], x[2], y[2]);
   line(x[2], y[2], x[0], y[0]);


   int s[2][2] = { sx, 0, 0, sy };
   int p[2][1];

   for (int i = 0; i < 3; i++)
   {
       p[0][0] = x[i];
       p[1][0] = y[i];

       findNewCoordinate(s, p);

       x[i] = p[0][0];
       y[i] = p[1][0];
   }
   setcolor(GREEN);
   line(x[0], y[0], x[1], y[1]);
   line(x[1], y[1], x[2], y[2]);
   line(x[2], y[2], x[0], y[0]);
}
```

```
int main()
{
   int x[3] , y[3] , sx, sy;
   printf("Enter Three Coordinates: ");
   for(int i = 0; i < 3; i++)
       scanf("%d%d",&x[i], &y[i]);
   getchar();
   printf("Enter Scaling Factor (Sx and Sy): ");
   scanf("%d%d",&sx,&sy);
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "C://TurboC3//BGI");
   scale(x, y, sx,sy);
   getch();
   closegraph();
   return 0;
}
```

Output:



Experiment 4: Line Clipping and Polygon Clipping

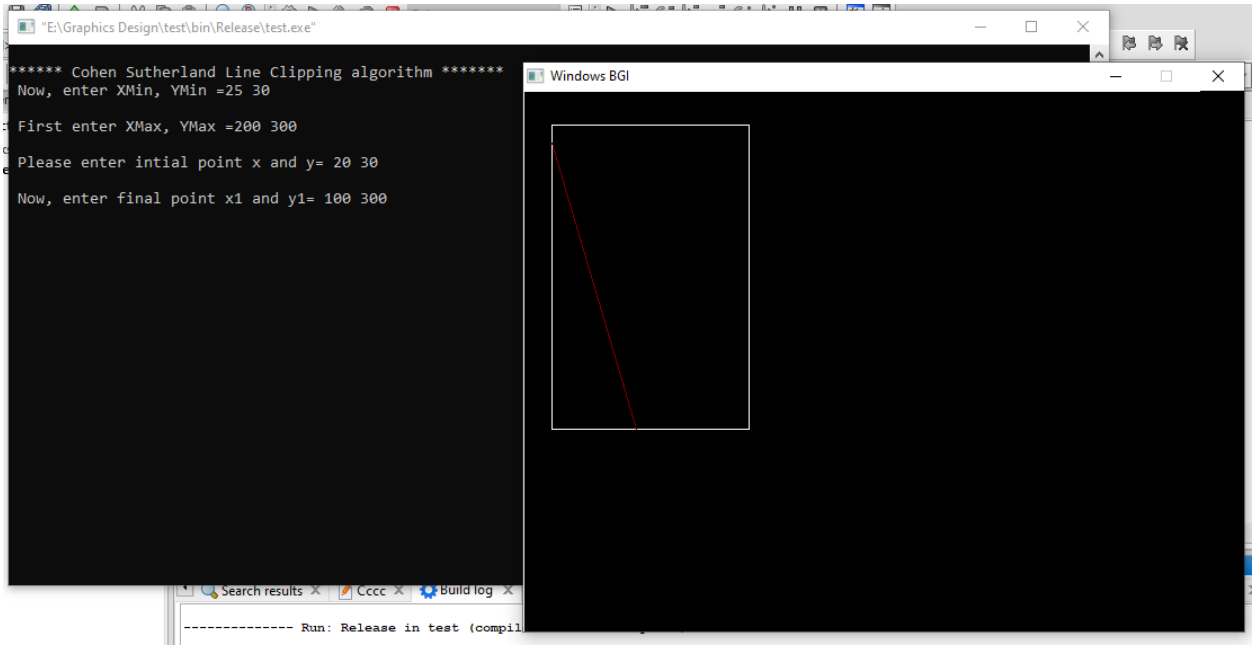
Line Clipping(Using Cohen Sutherland Algorithm):

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

int main()
{
    int rcode_begin[4]={0,0,0,0},rcode_end[4]={0,0,0,0},region_code[4];
    int W_xmax,W_ymax,W_xmin,W_ymin,flag=0;
    float slope;
    int x,y,x1,y1,i, xc,yc;
    int gr=DETECT,gm;
    initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
    printf("\n****** Cohen Sutherland Line Clipping algorithm *******");
    printf("\n Now, enter XMin, YMin =");

    scanf("%d %d",&W_xmin,&W_ymin);
    printf("\n First enter XMax, YMax =");
    scanf("%d %d",&W_xmax,&W_ymax);
    printf("\n Please enter intial point x and y= ");
    scanf("%d %d",&x,&y);
    printf("\n Now, enter final point x1 and y1= ");
    scanf("%d %d",&x1,&y1);
    cleardevice();

    rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
    line(x,y,x1,y1);
    line(0,0,600,0);
    line(0,0,0,600);

    if(y>W_ymax)  {
        rcode_begin[0]=1;     // Top
        flag=1 ;
    }
    if(y<W_ymin) {
        rcode_begin[1]=1;           // Bottom
        flag=1;
    }
    if(x>W_xmax)  {
        rcode_begin[2]=1;           // Right
        flag=1;
    }
```

```
if(x<W_xmin)   {
    rcode_begin[3]=1;           //Left
    flag=1;
}

//end point of Line

if(y1>W_ymax){
    rcode_end[0]=1;           // Top
    flag=1;
}

if(y1<W_ymin) {
    rcode_end[1]=1;           // Bottom
    flag=1;
}
if(x1>W_xmax){
    rcode_end[2]=1;           // Right
    flag=1;
}
if(x1<W_xmin){
    rcode_end[3]=1;           //Left
    flag=1;
}
if(flag==0)
    printf("No need of clipping as it is already in window");
flag=1;
for(i=0;i<4;i++){
    region_code[i]= rcode_begin[i] && rcode_end[i] ;
    if(region_code[i]==1)
        flag=0;
}
if(flag==0){
    printf("\n Line is completely outside the window");
}
else{
    slope=(float)(y1-y)/(x1-x);
    if(rcode_begin[2]==0 && rcode_begin[3]==1) {
        y=y+(float) (W_xmin-x)*slope ;
        x=W_xmin;
    }
    if(rcode_begin[2]==1 && rcode_begin[3]==0){
        y=y+(float) (W_xmax-x)*slope ;
```

```
            x=W_xmax;
        }
        if(rcode_begin[0]==1 && rcode_begin[1]==0){
            x=x+(float) (W_ymax-y)/slope ;
            y=W_ymax;
        }
        if(rcode_begin[0]==0 && rcode_begin[1]==1){
            x=x+(float) (W_ymin-y)/slope ;
            y=W_ymin;
        }
        // end points
        if(rcode_end[2]==0 && rcode_end[3]==1){
            y1=y1+(float) (W_xmin-x1)*slope ;
            x1=W_xmin;
        }
        if(rcode_end[2]==1 && rcode_end[3]==0){
            y1=y1+(float) (W_xmax-x1)*slope ;
            x1=W_xmax;
        }
        if(rcode_end[0]==1 && rcode_end[1]==0){
            x1=x1+(float) (W_ymax-y1)/slope ;
            y1=W_ymax;
        }
        if(rcode_end[0]==0 && rcode_end[1]==1){
            x1=x1+(float) (W_ymin-y1)/slope ;
            y1=W_ymin;
        }
    }
    delay(1000);
    clearviewport();
    rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
    line(0,0,600,0);
    line(0,0,0,600);
    setcolor(RED);
    line(x,y,x1,y1);
    getch();
    closegraph();
    return 0;
}
```

Output:



Polygon Clipping(Using Sutherland Hodgeman Algorithm):

```
#include<stdio.h>
#include<graphics.h>
#include<stdlib.h>
int main()
{
   int gd,n,*x,i,k=0,wx1=220,wy1=140,wx2=420,wy2=140,wx3=420,wy3=340;
   int wx4=220,wy4=340;
window int w[]={220,140,420,140,420,340,220,340,220,140};//array for drawing
   int gr=DETECT,gm;
   initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
   printf("Window:-");
   setcolor(RED); //red colored window
   drawpoly(5,w); //window drawn
   printf("Enter the no. of vertices of polygon: ");
   scanf("%d",&n);
   x = (int*)malloc(n*2+1);
   printf("Enter the coordinates of points:\n");
   k=0;
   for(i=0;i<n*2;i+=2) //reading vertices of polygon
   {
          printf("(x%d,y%d): ",k,k);
          scanf("%d,%d",&x[i],&x[i+1]);
          k++;
   }
   x[n*2]=x[0];
   x[n*2+1]=x[1];
   setcolor(WHITE);
   drawpoly(n+1,x);
   printf("\nPress a button to clip a polygon..");
```

```
    getch();
    setcolor(RED);
    drawpoly(5,w);
    setfillstyle(SOLID_FILL,BLACK);
    floodfill(2,2,RED);
    //gotoxy(1,1); //bringing cursor at starting position
    printf("\nThis is the clipped polygon..");
    getch();
    cleardevice();
    closegraph();
}
```

Experiment 5: Bezier Curves

Code:

```
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<stdio.h>
int main()
{
    int x[4],y[4],i;
    double put_x,put_y,t;
    int gr=DETECT,gm;
    initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
    printf("\n****** Bezier Curve ***********");
    printf("\n Please enter x and y coordinates ");
    for(i=0;i<4;i++)
    {
        scanf("%d%d",&x[i],&y[i]);
        putpixel(x[i],y[i],3);
    }

    for(t=0.0;t<=1.0;t=t+0.001)
    {
        put_x = pow(1-t,3)*x[0] + 3*t*pow(1-t,2)*x[1] +
         3*t*t*(1-t)*x[2] + pow(t,3)*x[3];

        put_y =  pow(1-t,3)*y[0] + 3*t*pow(1-t,2)*y[1] +
         3*t*t*(1-t)*y[2] + pow(t,3)*y[3];

        putpixel(put_x,put_y, WHITE);
    }

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

Output:


"E:\Graphics Design\test\bin\Release\test.exe"
****** Bezier Curve ***********
Please enter x and y coordinates 100 300 150 50 300 68 400 65
Windows BGI

Experiment 6: Koch Curve(Fractal)

Code:

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

void koch(int x1, int y1, int x2, int y2, int it)
{
 float angle = 60*M_PI/180;
 int x3 = (2*x1+x2)/3;
 int y3 = (2*y1+y2)/3;

 int x4 = (x1+2*x2)/3;
 int y4 = (y1+2*y2)/3;

 int x = x3 + (x4-x3)*cos(angle)+(y4-y3)*sin(angle);
 int y = y3 - (x4-x3)*sin(angle)+(y4-y3)*cos(angle);

 if(it > 0)
 {
  koch(x1, y1, x3, y3, it-1);
  koch(x3, y3, x, y, it-1);
  koch(x, y, x4, y4, it-1);
  koch(x4, y4, x2, y2, it-1);
 }
 else
 {

  line(x1, y1, x3, y3);
  line(x3, y3, x, y);
  line(x, y, x4, y4);
  line(x4, y4, x2, y2);
 }
}

int main(void)
{
 int gd = DETECT, gm;
 initgraph(&gd, &gm, "C:\\TC\\BGI");
 int x1 = 100, y1 = 100, x2 = 400, y2 = 400;
 koch(x1, y1, x2, y2, 4);
 getch();
 return 0;
}
```

Output: