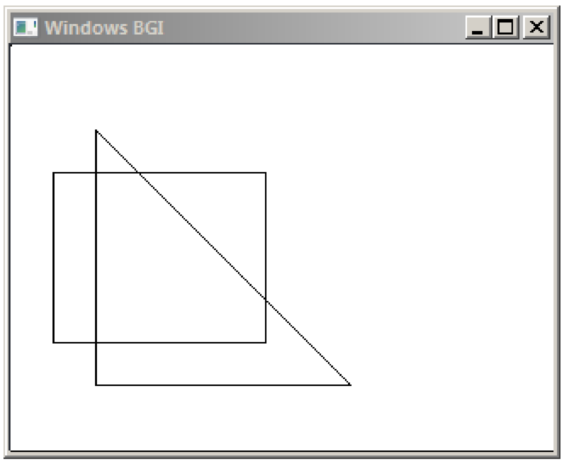
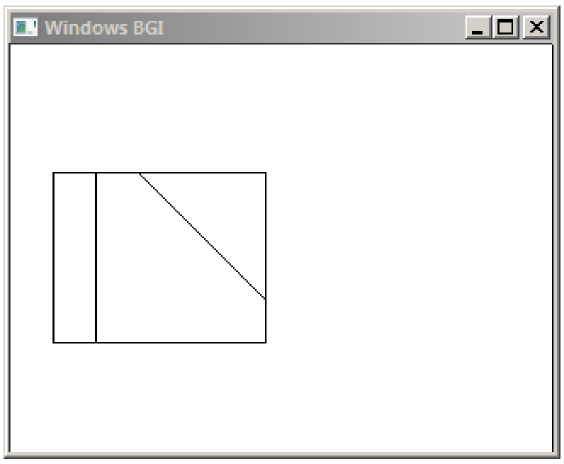
```
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <process.h>

#define TRUE 1
#define FALSE 0

typedef unsigned int outcode;
outcode CompOutCode(double x, double y);
enum { TOP = 0x1, BOTTOM = 0x2, RIGHT = 0x4, LEFT = 0x8 };
float xmin, xmax, ymin, ymax;

void clip(double x0, double y0, double x1, double y1)
{
        outcode outcode0, outcode1, outcodeOut;
        int accept = FALSE, done = FALSE;
        outcode0 = CompOutCode(x0, y0);
        outcode1 = CompOutCode(x1, y1);
        do
        {
                if (!(outcode0 | outcode1))
                {
                        accept = TRUE;
                        done = TRUE;
                }
                else if (outcode0 & outcode1)
                        done = TRUE;
                else
                {
                        double x, y;
                        outcodeOut = outcode0 ? outcode0 : outcode1;
                        if (outcodeOut & TOP)
                        {
                                x = x0 + (x1 - x0) * (ymax - y0) / (y1 - y0);
                                y = ymax;
                        }
                        else if (outcodeOut & BOTTOM)
                        {
                                x = x0 + (x1 - x0) * (ymin - y0) / (y1 - y0);
                                y = ymin;
                        }
                        else if (outcodeOut & RIGHT)
                        {
                                y = y0 + (y1 - y0) * (xmax - x0) / (x1 - x0);
                                x = xmax;
                        }
                        else
                        {
                                y = y0 + (y1 - y0) * (xmin - x0) / (x1 - x0);
                                x = xmin;
                        }
                        if (outcodeOut == outcode0)
                        {
                                x0 = x;
                                y0 = y;
                                outcode0 = CompOutCode(x0, y0);
                        }
                        else
                        {
                                x1 = x;
                                y1 = y;
                                outcode1 = CompOutCode(x1, y1);
                        }
                }
        } while (done == FALSE);
        if (accept)
                line(x0, y0, x1, y1);

        rectangle(xmin, ymin, xmax, ymax);
        }

outcode CompOutCode(double x, double y)
```

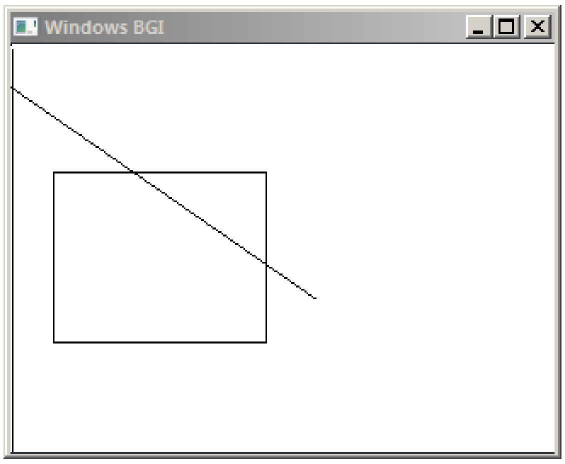
```
{
        outcode code = 0;
        if (y > ymax)
                code |= TOP;
        else if (y < ymin)
                code |= BOTTOM;
        if (x > xmax)
                code |= RIGHT;
        else if (x < xmin)
                code |= LEFT;
        return code;
}

int main()
{
        double x1, y1, x2, y2;
        int n, poly[14], i;

        printf("Number of vertices: ");
        scanf("%d", &n);
        printf("Enter vertices:\n");

        for (i = 0; i < 2 * n; i++)
                scanf("%d", &poly[i]);

        poly[2 * n] = poly[0];
        poly[2 * n + 1] = poly[1];

        printf("Window coordinates (min, max): ");
        scanf("%f%f%f%f", &xmin, &ymin, &xmax, &ymax);

        initwindow(640, 480);

        drawpoly(n + 1, poly);
        rectangle(xmin, ymin, xmax, ymax);

        while( !kbhit() );
        cleardevice();

        for (i = 0; i < n; i++)
                clip(poly[2 * i], poly[(2 * i) + 1], poly[(2 * i) + 2], poly[(2 * i) + 3]);

        while( !kbhit() );
        return EXIT_SUCCESS;
}
```

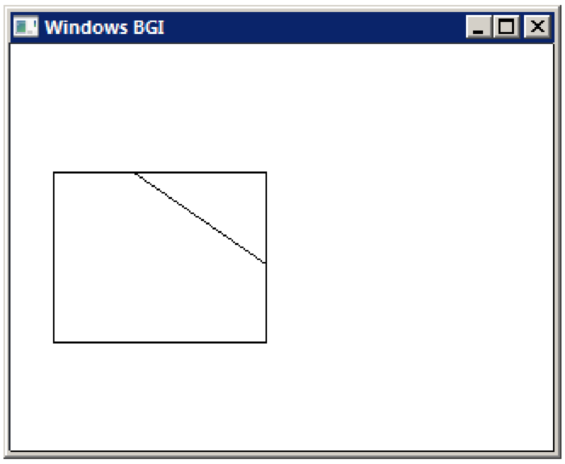




```
Number of vertices: 3
Enter vertices:
50 50 50 200 200 200
Window coordinates (min, max): 25 75 150 175
```

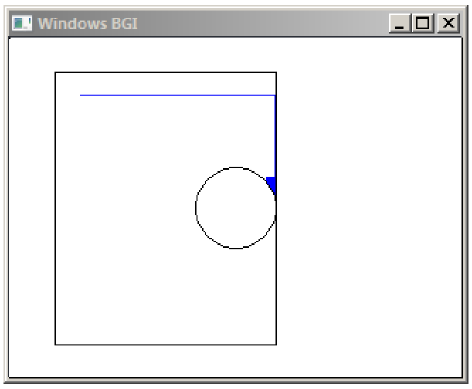
```c
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <conio.h>

#define MAX 20
#define TRUE 1
#define FALSE 0

int top=1, bottom=2, right=4, left=8;
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);
}

int main()
{
        int n, i, j, ln[1][4], clip[4];

        printf("Window coordinates (min, max): ");
        scanf("%d %d %d %d", &clip[0], &clip[1], &clip[2], &clip[3]);

        printf("Line coordinates: \n");
        scanf("%d %d %d %d", &ln[0][0], &ln[0][1], &ln[0][2], &ln[0][3]);

        initwindow(320, 240);

        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();

        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();
        }
        return 0;
}
```

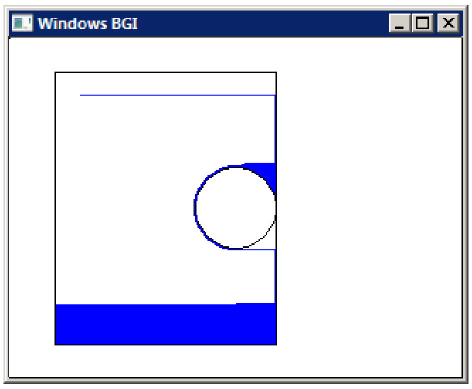


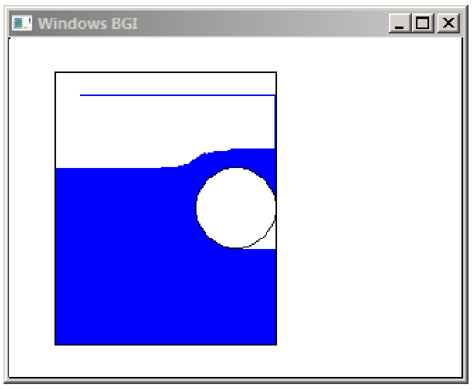


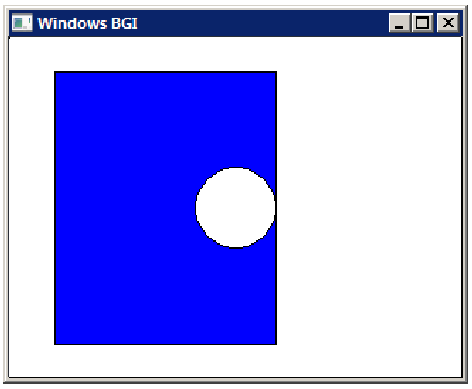
```
Window coordinates (min, max): 25 75 150 175
Line coordinates:
0 25 180 150
```

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

void scanline(int, int, int, int);
void scanline(int x, int y, int fill_color, int border)
{
        putpixel(x,y,fill_color);

        if(getpixel(x+1,y) != border && getpixel(x+1,y) != fill_color)
                scanline(x+1,y,fill_color,border);

        if(getpixel(x,y+1) != border && getpixel(x,y+1) != fill_color)
                scanline(x,y+1,fill_color,border);

        if(getpixel(x-1,y) != border && getpixel(x-1,y) != fill_color)
                scanline(x-1,y,fill_color,border);

        if(getpixel(x,y-1) != border && getpixel(x,y-1) != fill_color)
                scanline(x,y-1,fill_color,border);

        return;
}

int main()
{
        initwindow(320,240);

        rectangle(32, 24, 188, 216);
        circle(160, 120, 29);

        scanline(50, 40, BLUE, WHITE);

        while(!kbhit())
                delay(50);

        return EXIT_SUCCESS;
}
```


Windows BGI


Windows BGI


Windows BGI


Windows BGI