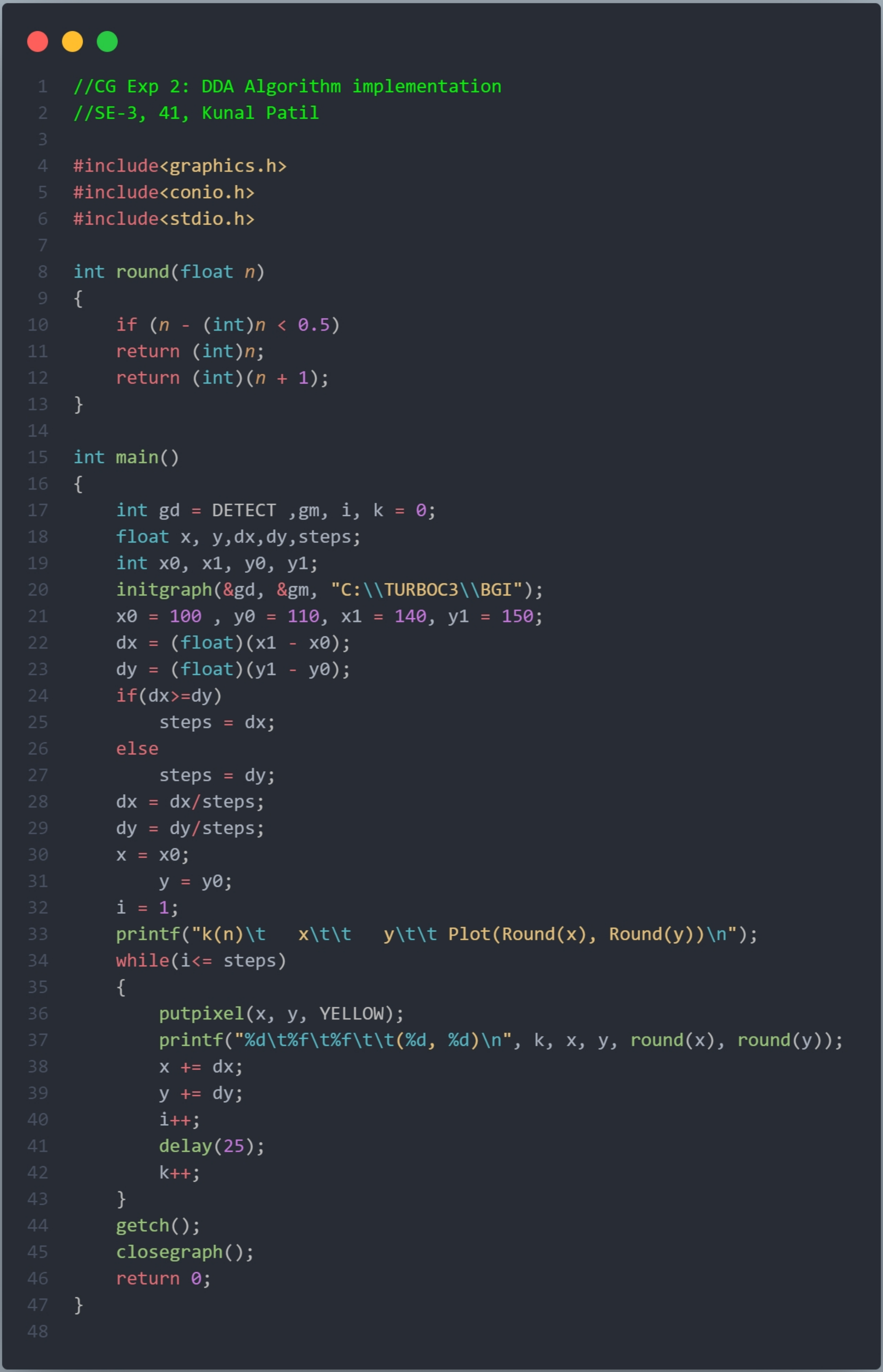
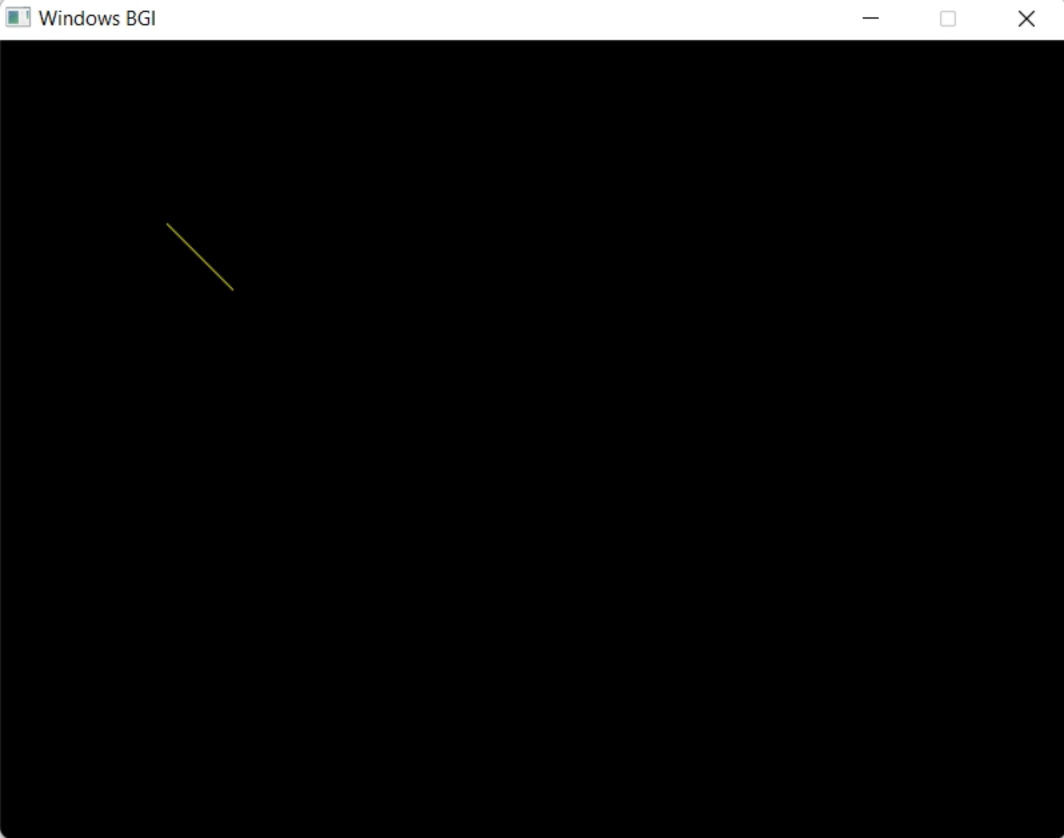


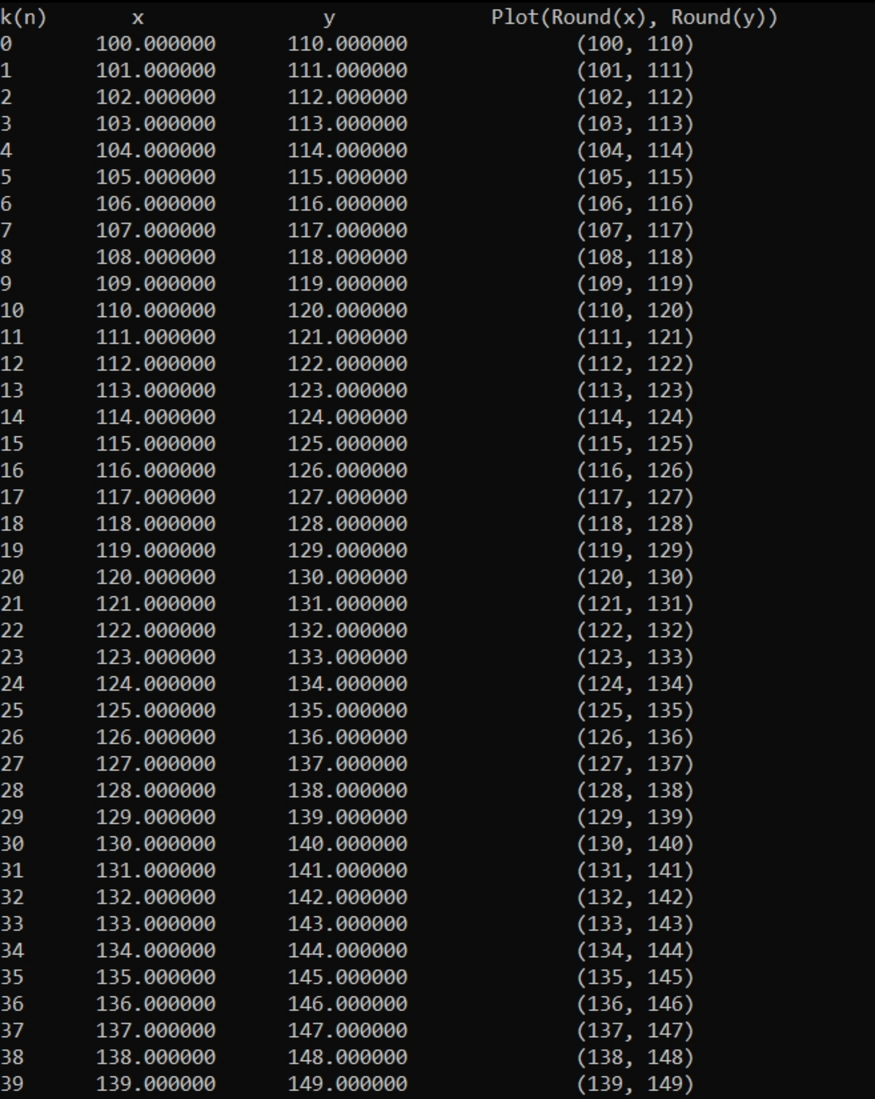
```c
//CG Exp 2: DDA Algorithm implementation
//SE-3, 41, Kunal Patil

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

int round(float n)
{
    if (n - (int)n < 0.5)
    return (int)n;
    return (int)(n + 1);
}

int main()
{
    int gd = DETECT ,gm, i, k = 0;
    float x, y,dx,dy,steps;
    int x0, x1, y0, y1;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    x0 = 100 , y0 = 110, x1 = 140, y1 = 150;
    dx = (float)(x1 - x0);
    dy = (float)(y1 - y0);
    if(dx>=dy)
        steps = dx;
    else
        steps = dy;
    dx = dx/steps;
    dy = dy/steps;
    x = x0;
        y = y0;
    i = 1;
    printf("k(n)\t   x\t\t   y\t\t Plot(Round(x), Round(y))\n");
    while(i<= steps)
    {
        putpixel(x, y, YELLOW);
        printf("%d\t%f\t%f\t\t(%d, %d)\n", k, x, y, round(x), round(y));
        x += dx;
        y += dy;
        i++;
        delay(25);
        k++;
    }
    getch();
    closegraph();
    return 0;
}
```


Windows BGI



```
k(n)       x              y             Plot(Round(x), Round(y))
0      100.000000     110.000000             (100, 110)
1      101.000000     111.000000             (101, 111)
2      102.000000     112.000000             (102, 112)
3      103.000000     113.000000             (103, 113)
4      104.000000     114.000000             (104, 114)
5      105.000000     115.000000             (105, 115)
6      106.000000     116.000000             (106, 116)
7      107.000000     117.000000             (107, 117)
8      108.000000     118.000000             (108, 118)
9      109.000000     119.000000             (109, 119)
10     110.000000     120.000000             (110, 120)
11     111.000000     121.000000             (111, 121)
12     112.000000     122.000000             (112, 122)
13     113.000000     123.000000             (113, 123)
14     114.000000     124.000000             (114, 124)
15     115.000000     125.000000             (115, 125)
16     116.000000     126.000000             (116, 126)
17     117.000000     127.000000             (117, 127)
18     118.000000     128.000000             (118, 128)
19     119.000000     129.000000             (119, 129)
20     120.000000     130.000000             (120, 130)
21     121.000000     131.000000             (121, 131)
22     122.000000     132.000000             (122, 132)
23     123.000000     133.000000             (123, 133)
24     124.000000     134.000000             (124, 134)
25     125.000000     135.000000             (125, 135)
26     126.000000     136.000000             (126, 136)
27     127.000000     137.000000             (127, 137)
28     128.000000     138.000000             (128, 138)
29     129.000000     139.000000             (129, 139)
30     130.000000     140.000000             (130, 140)
31     131.000000     141.000000             (131, 141)
32     132.000000     142.000000             (132, 142)
33     133.000000     143.000000             (133, 143)
34     134.000000     144.000000             (134, 144)
35     135.000000     145.000000             (135, 145)
36     136.000000     146.000000             (136, 146)
37     137.000000     147.000000             (137, 147)
38     138.000000     148.000000             (138, 148)
39     139.000000     149.000000             (139, 149)
```