Batch: D-2 **Roll No.:** 16010122151

Experiment No. 03

Title: Drawing of line using computer graphics.

Aim: Generate the line using computer graphics program

Objectives:

1. Visit the Following link and perform the Vlab Experiment and provide the screenshots. https://cse18-iiith.vlabs.ac.in/exp/coordinate-systems/pretest.html
2. Implement the Digital Differential Analyser (DDA) Line Drawing Algorithm.
3. Implement Bresenham Line Drawing Algorithm

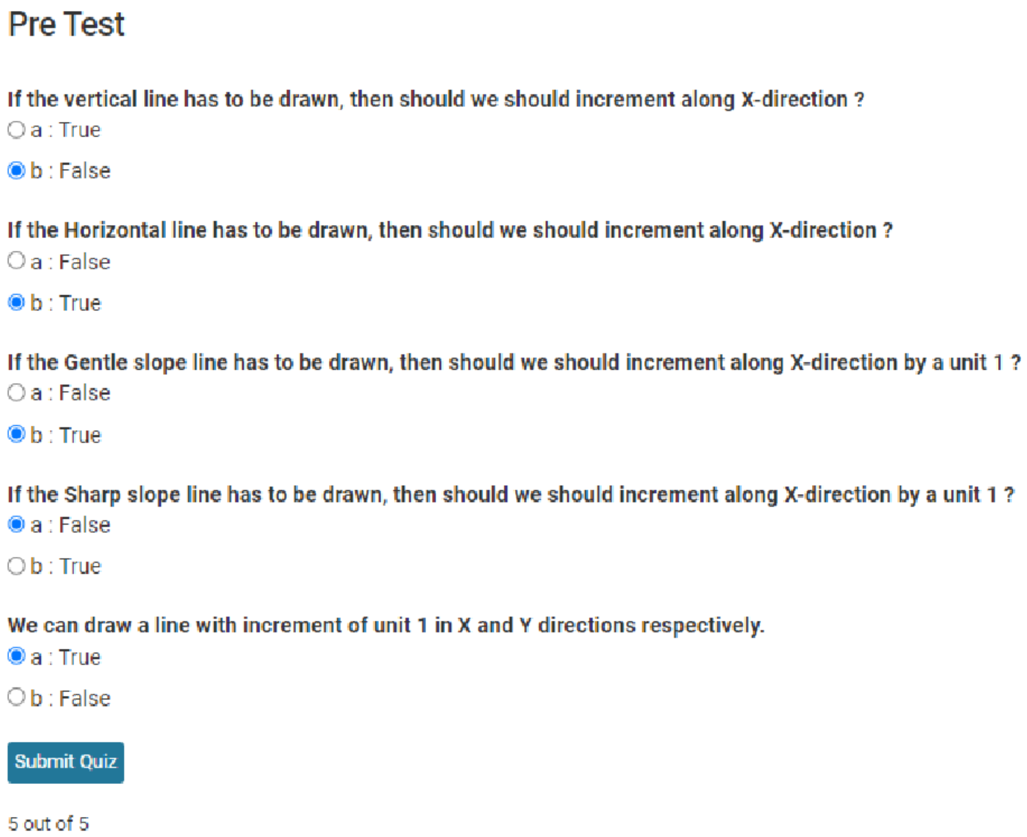
Expected OUTCOME of Experiment:

Understand the basic concepts of computer graphics and OpenGL

Books/ Journals/ Websites referred:

Screenshots from VLab

Algorithm 1: DDA



Pre Test

If the vertical line has to be drawn, then should we should increment along X-direction ?

○ a : True

● b : False

If the Horizontal line has to be drawn, then should we should increment along X-direction ?

○ a : False

● b : True

If the Gentle slope line has to be drawn, then should we should increment along X-direction by a unit 1 ?

○ a : False

● b : True

If the Sharp slope line has to be drawn, then should we should increment along X-direction by a unit 1 ?

● a : False

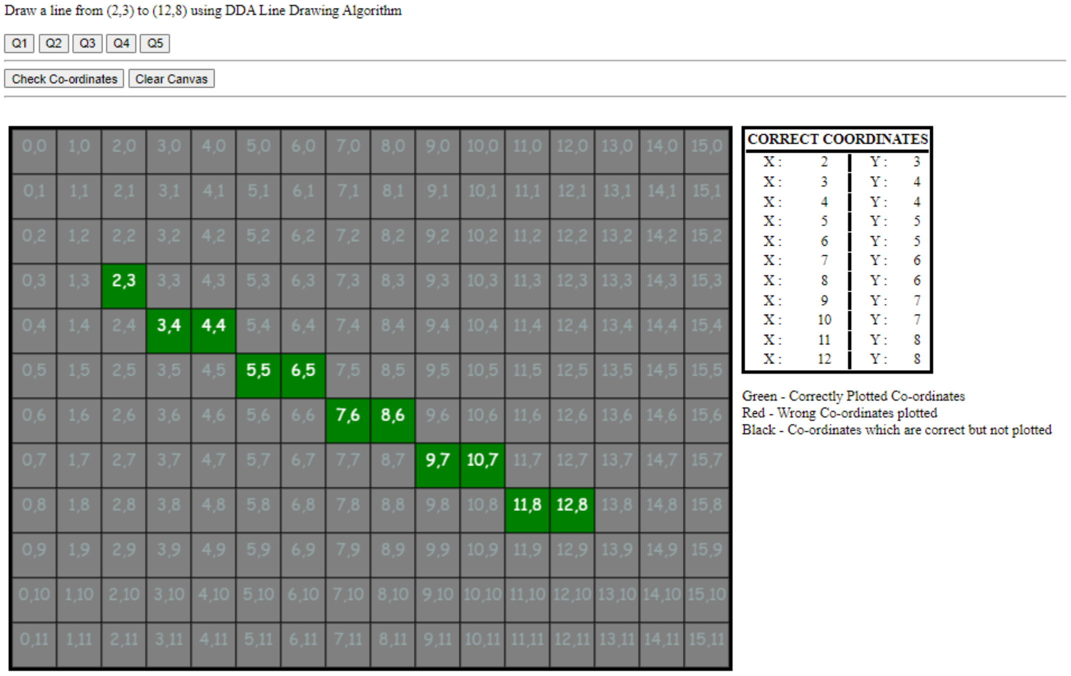
○ b : True

We can draw a line with increment of unit 1 in X and Y directions respectively.

● a : True

○ b : False

5 out of 5

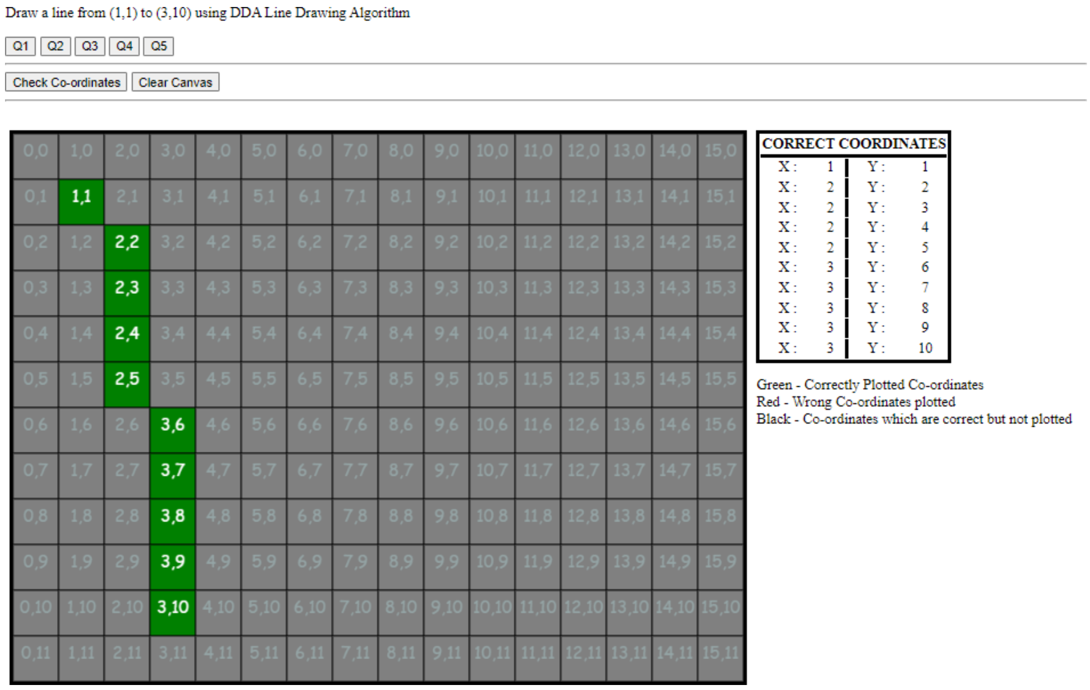


Draw a line from (2,3) to (12,8) using DDA Line Drawing Algorithm

0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0	12,0	13,0	14,0	15,0
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1	11,1	12,1	13,1	14,1	15,1
0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2	10,2	11,2	12,2	13,2	14,2	15,2
0,3	1,3	**2,3**	3,3	4,3	5,3	6,3	7,3	8,3	9,3	10,3	11,3	12,3	13,3	14,3	15,3
0,4	1,4	2,4	**3,4**	**4,4**	5,4	6,4	7,4	8,4	9,4	10,4	11,4	12,4	13,4	14,4	15,4
0,5	1,5	2,5	3,5	4,5	**5,5**	**6,5**	7,5	8,5	9,5	10,5	11,5	12,5	13,5	14,5	15,5
0,6	1,6	2,6	3,6	4,6	5,6	6,6	**7,6**	**8,6**	9,6	10,6	11,6	12,6	13,6	14,6	15,6
0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	**9,7**	**10,7**	11,7	12,7	13,7	14,7	15,7
0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8	10,8	**11,8**	**12,8**	13,8	14,8	15,8
0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9	10,9	11,9	12,9	13,9	14,9	15,9
0,10	1,10	2,10	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10,10	11,10	12,10	13,10	14,10	15,10
0,11	1,11	2,11	3,11	4,11	5,11	6,11	7,11	8,11	9,11	10,11	11,11	12,11	13,11	14,11	15,11

CORRECT COORDINATES			
X :	2	Y :	3
X :	3	Y :	4
X :	4	Y :	4
X :	5	Y :	5
X :	6	Y :	5
X :	7	Y :	6
X :	8	Y :	6
X :	9	Y :	7
X :	10	Y :	7
X :	11	Y :	8
X :	12	Y :	8

Green - Correctly Plotted Co-ordinates
Red - Wrong Co-ordinates plotted
Black - Co-ordinates which are correct but not plotted

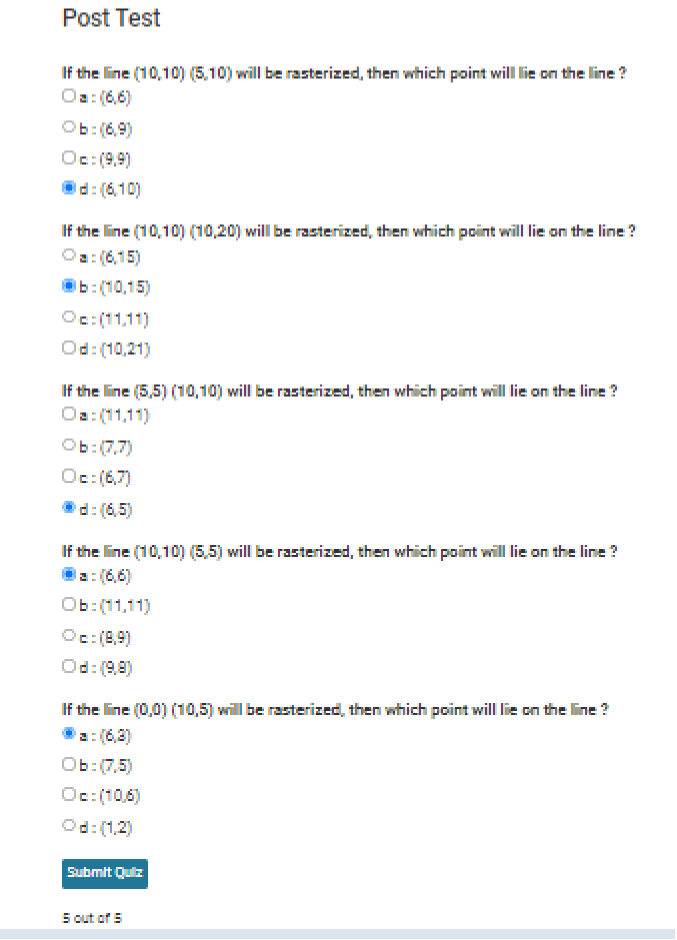


Draw a line from (1,1) to (3,10) using DDA Line Drawing Algorithm

0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0	12,0	13,0	14,0	15,0
0,1	**1,1**	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1	11,1	12,1	13,1	14,1	15,1
0,2	1,2	**2,2**	3,2	4,2	5,2	6,2	7,2	8,2	9,2	10,2	11,2	12,2	13,2	14,2	15,2
0,3	1,3	**2,3**	3,3	4,3	5,3	6,3	7,3	8,3	9,3	10,3	11,3	12,3	13,3	14,3	15,3
0,4	1,4	**2,4**	3,4	4,4	5,4	6,4	7,4	8,4	9,4	10,4	11,4	12,4	13,4	14,4	15,4
0,5	1,5	**2,5**	3,5	4,5	5,5	6,5	7,5	8,5	9,5	10,5	11,5	12,5	13,5	14,5	15,5
0,6	1,6	2,6	**3,6**	4,6	5,6	6,6	7,6	8,6	9,6	10,6	11,6	12,6	13,6	14,6	15,6
0,7	1,7	2,7	**3,7**	4,7	5,7	6,7	7,7	8,7	9,7	10,7	11,7	12,7	13,7	14,7	15,7
0,8	1,8	2,8	**3,8**	4,8	5,8	6,8	7,8	8,8	9,8	10,8	11,8	12,8	13,8	14,8	15,8
0,9	1,9	2,9	**3,9**	4,9	5,9	6,9	7,9	8,9	9,9	10,9	11,9	12,9	13,9	14,9	15,9
0,10	1,10	2,10	**3,10**	4,10	5,10	6,10	7,10	8,10	9,10	10,10	11,10	12,10	13,10	14,10	15,10
0,11	1,11	2,11	3,11	4,11	5,11	6,11	7,11	8,11	9,11	10,11	11,11	12,11	13,11	14,11	15,11

CORRECT COORDINATES			
X :	1	Y :	1
X :	2	Y :	2
X :	2	Y :	3
X :	2	Y :	4
X :	2	Y :	5
X :	3	Y :	6
X :	3	Y :	7
X :	3	Y :	8
X :	3	Y :	9
X :	3	Y :	10

Green - Correctly Plotted Co-ordinates
Red - Wrong Co-ordinates plotted
Black - Co-ordinates which are correct but not plotted



Post Test

If the line (10,10) (5,10) will be rasterized, then which point will lie on the line ?

○ a : (6,6)

○ b : (6,9)

○ c : (9,9)

● d : (6,10)

If the line (10,10) (10,20) will be rasterized, then which point will lie on the line ?

○ a : (6,15)

● b : (10,15)

○ c : (11,11)

○ d : (10,21)

If the line (5,5) (10,10) will be rasterized, then which point will lie on the line ?

○ a : (11,11)

○ b : (7,7)

○ c : (6,7)

● d : (6,5)

If the line (10,10) (5,5) will be rasterized, then which point will lie on the line ?

● a : (6,6)

○ b : (11,11)

○ c : (8,9)

○ d : (9,8)

If the line (0,0) (10,5) will be rasterized, then which point will lie on the line ?

● a : (6,3)

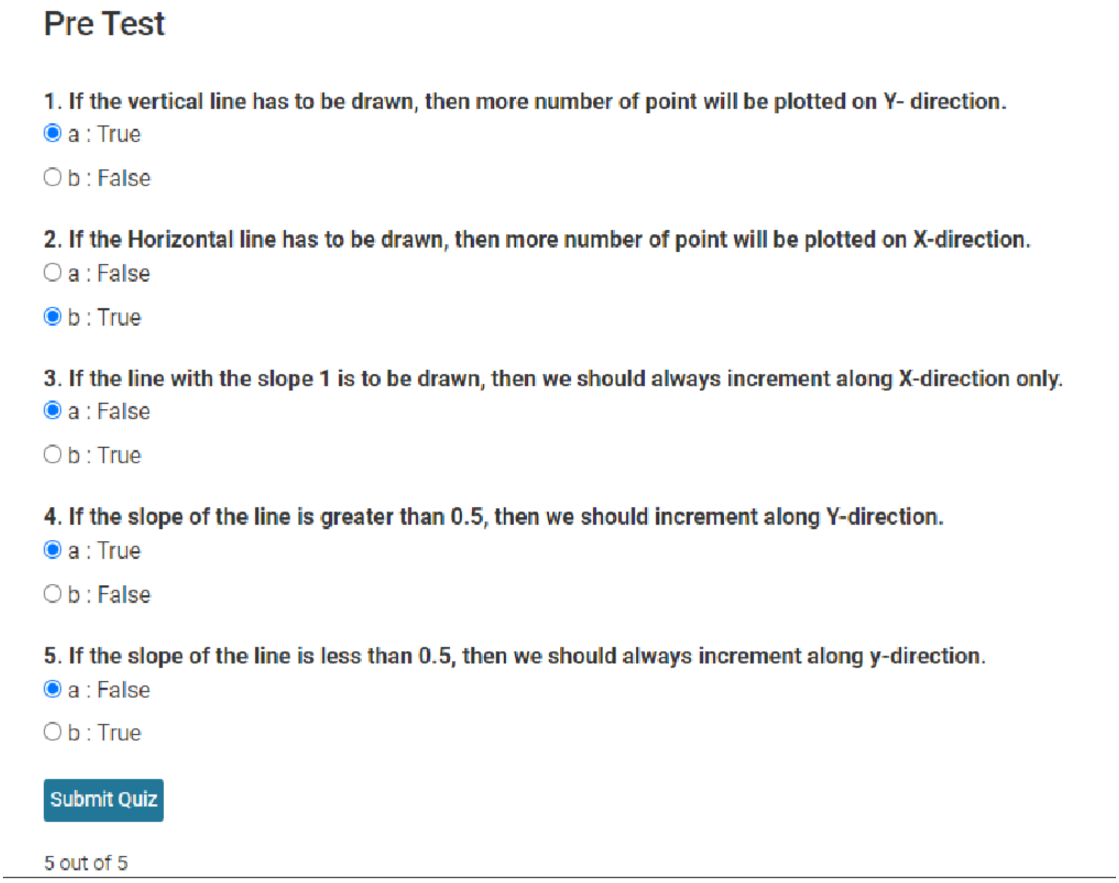
○ b : (7,5)

○ c : (10,6)

○ d : (1,2)

5 out of 5

Algorithm: 2 : Bresenham’s Line Drawing Algorithm



Pre Test

1. If the vertical line has to be drawn, then more number of point will be plotted on Y- direction.

● a : True

○ b : False

2. If the Horizontal line has to be drawn, then more number of point will be plotted on X-direction.

○ a : False

● b : True

3. If the line with the slope 1 is to be drawn, then we should always increment along X-direction only.

● a : False

○ b : True

4. If the slope of the line is greater than 0.5, then we should increment along Y-direction.

● a : True

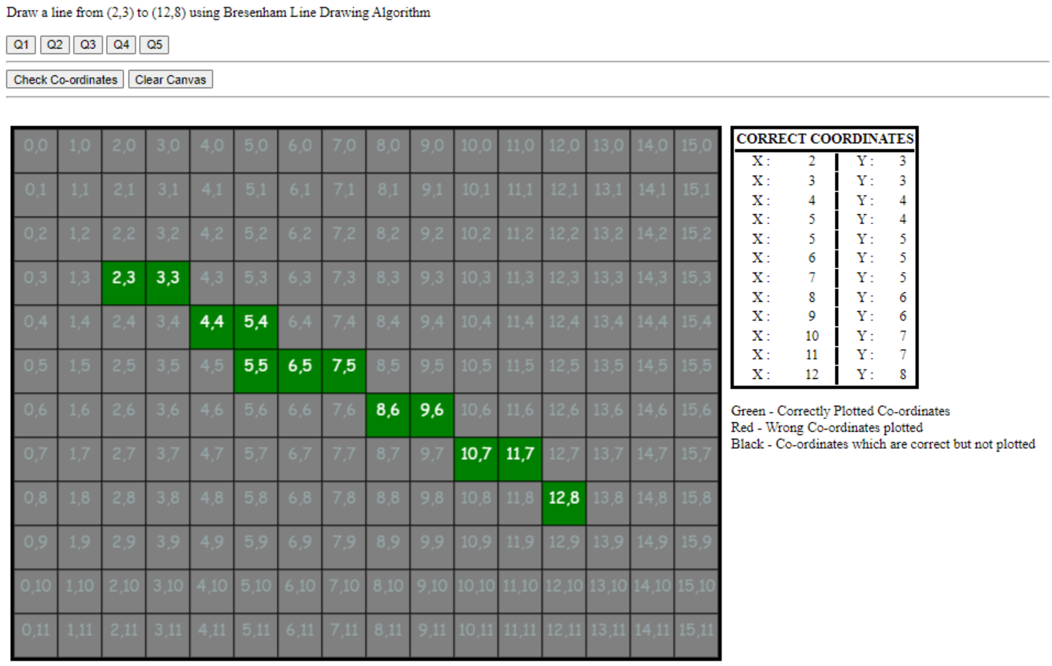
○ b : False

5. If the slope of the line is less than 0.5, then we should always increment along y-direction.

● a : False

○ b : True

5 out of 5



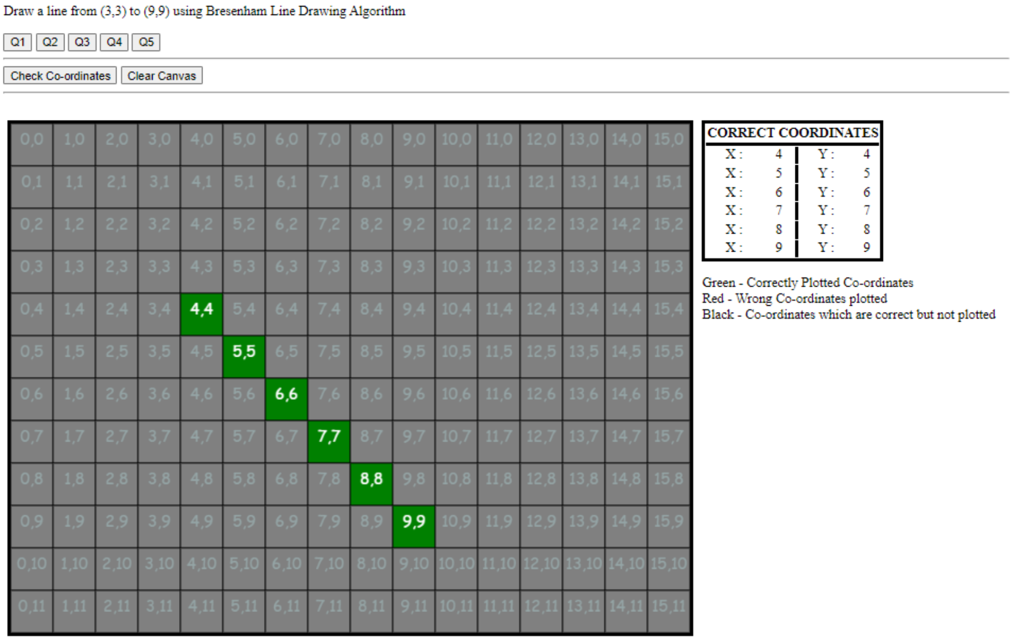
Draw a line from (2,3) to (12,8) using Bresenham Line Drawing Algorithm

0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0	12,0	13,0	14,0	15,0
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1	11,1	12,1	13,1	14,1	15,1
0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2	10,2	11,2	12,2	13,2	14,2	15,2
0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3	10,3	11,3	12,3	13,3	14,3	15,3
0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4	10,4	11,4	12,4	13,4	14,4	15,4
0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5	10,5	11,5	12,5	13,5	14,5	15,5
0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6	10,6	11,6	12,6	13,6	14,6	15,6
0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7	10,7	11,7	12,7	13,7	14,7	15,7
0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8	10,8	11,8	12,8	13,8	14,8	15,8
0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9	10,9	11,9	12,9	13,9	14,9	15,9
0,10	1,10	2,10	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10,10	11,10	12,10	13,10	14,10	15,10
0,11	1,11	2,11	3,11	4,11	5,11	6,11	7,11	8,11	9,11	10,11	11,11	12,11	13,11	14,11	15,11

CORRECT COORDINATES

X :	2	Y :	3
X :	3	Y :	3
X :	4	Y :	4
X :	5	Y :	4
X :	5	Y :	5
X :	6	Y :	5
X :	7	Y :	5
X :	8	Y :	6
X :	9	Y :	6
X :	10	Y :	7
X :	11	Y :	7
X :	12	Y :	8

Green - Correctly Plotted Co-ordinates
Red - Wrong Co-ordinates plotted
Black - Co-ordinates which are correct but not plotted



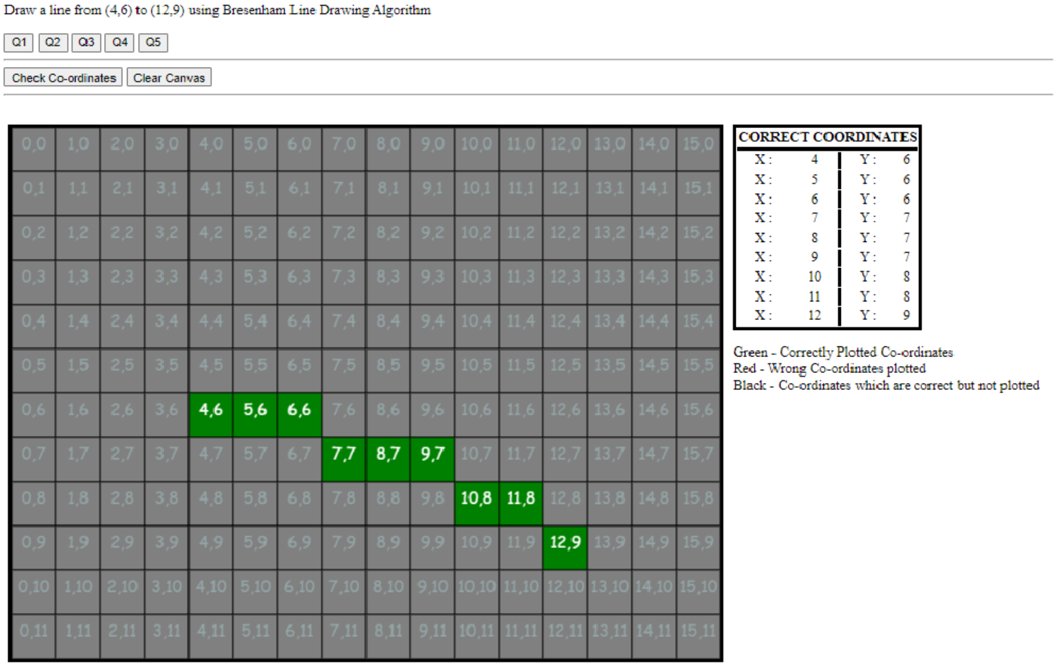
Draw a line from (3,3) to (9,9) using Bresenham Line Drawing Algorithm

0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0	12,0	13,0	14,0	15,0
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1	11,1	12,1	13,1	14,1	15,1
0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2	10,2	11,2	12,2	13,2	14,2	15,2
0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3	10,3	11,3	12,3	13,3	14,3	15,3
0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4	10,4	11,4	12,4	13,4	14,4	15,4
0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5	10,5	11,5	12,5	13,5	14,5	15,5
0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6	10,6	11,6	12,6	13,6	14,6	15,6
0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7	10,7	11,7	12,7	13,7	14,7	15,7
0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8	10,8	11,8	12,8	13,8	14,8	15,8
0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9	10,9	11,9	12,9	13,9	14,9	15,9
0,10	1,10	2,10	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10,10	11,10	12,10	13,10	14,10	15,10
0,11	1,11	2,11	3,11	4,11	5,11	6,11	7,11	8,11	9,11	10,11	11,11	12,11	13,11	14,11	15,11

CORRECT COORDINATES

X :	4	Y :	4
X :	5	Y :	5
X :	6	Y :	6
X :	7	Y :	7
X :	8	Y :	8
X :	9	Y :	9

Green - Correctly Plotted Co-ordinates
Red - Wrong Co-ordinates plotted
Black - Co-ordinates which are correct but not plotted



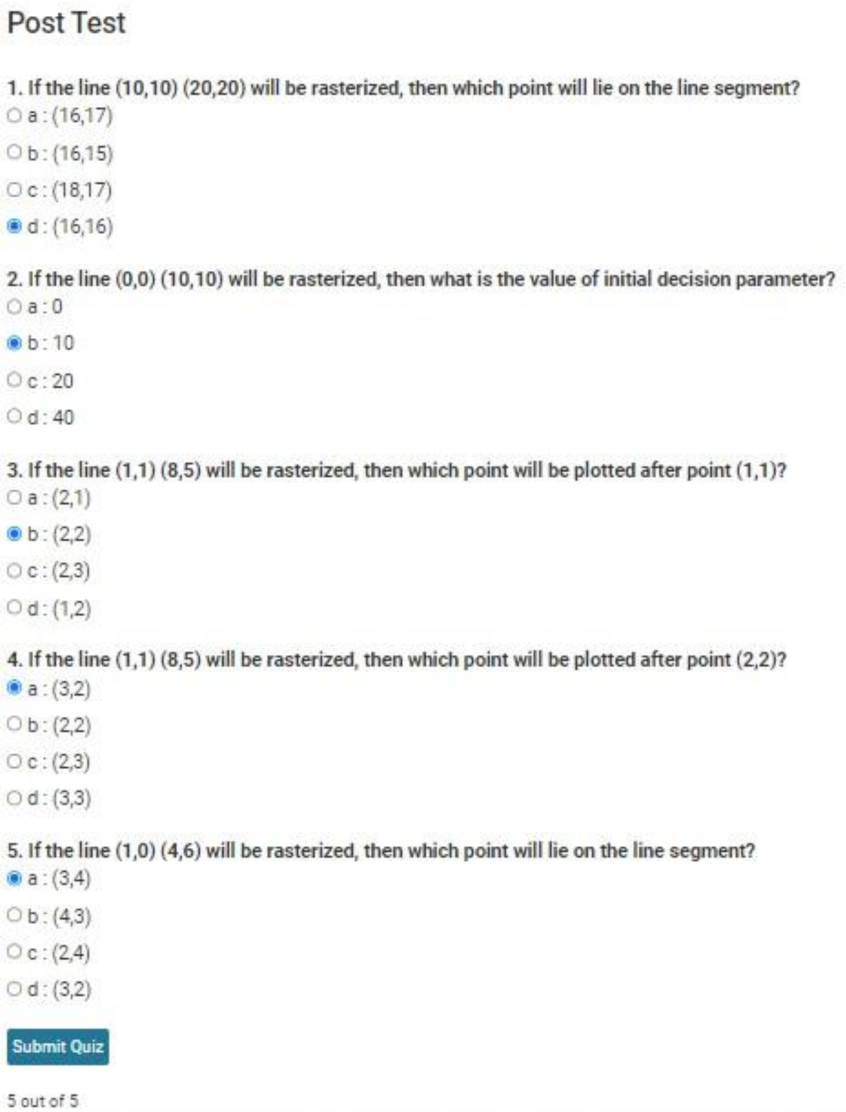
Draw a line from (4,6) to (12,9) using Bresenham Line Drawing Algorithm

0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0	12,0	13,0	14,0	15,0
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1	11,1	12,1	13,1	14,1	15,1
0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2	10,2	11,2	12,2	13,2	14,2	15,2
0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3	10,3	11,3	12,3	13,3	14,3	15,3
0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4	10,4	11,4	12,4	13,4	14,4	15,4
0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5	10,5	11,5	12,5	13,5	14,5	15,5
0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6	10,6	11,6	12,6	13,6	14,6	15,6
0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7	10,7	11,7	12,7	13,7	14,7	15,7
0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8	10,8	11,8	12,8	13,8	14,8	15,8
0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9	10,9	11,9	12,9	13,9	14,9	15,9
0,10	1,10	2,10	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10,10	11,10	12,10	13,10	14,10	15,10
0,11	1,11	2,11	3,11	4,11	5,11	6,11	7,11	8,11	9,11	10,11	11,11	12,11	13,11	14,11	15,11

CORRECT COORDINATES

X :	4	Y :	6
X :	5	Y :	6
X :	6	Y :	6
X :	7	Y :	7
X :	8	Y :	7
X :	9	Y :	7
X :	10	Y :	8
X :	11	Y :	8
X :	12	Y :	9

Green - Correctly Plotted Co-ordinates
Red - Wrong Co-ordinates plotted
Black - Co-ordinates which are correct but not plotted



Post Test

1. If the line (10,10) (20,20) will be rasterized, then which point will lie on the line segment?

○ a : (16,17)

○ b : (16,15)

○ c : (18,17)

● d : (16,16)

2. If the line (0,0) (10,10) will be rasterized, then what is the value of initial decision parameter?

○ a : 0

● b : 10

○ c : 20

○ d : 40

3. If the line (1,1) (8,5) will be rasterized, then which point will be plotted after point (1,1)?

○ a : (2,1)

● b : (2,2)

○ c : (2,3)

○ d : (1,2)

4. If the line (1,1) (8,5) will be rasterized, then which point will be plotted after point (2,2)?

● a : (3,2)

○ b : (2,2)

○ c : (2,3)

○ d : (3,3)

5. If the line (1,0) (4,6) will be rasterized, then which point will lie on the line segment?

● a : (3,4)

○ b : (4,3)

○ c : (2,4)

○ d : (3,2)

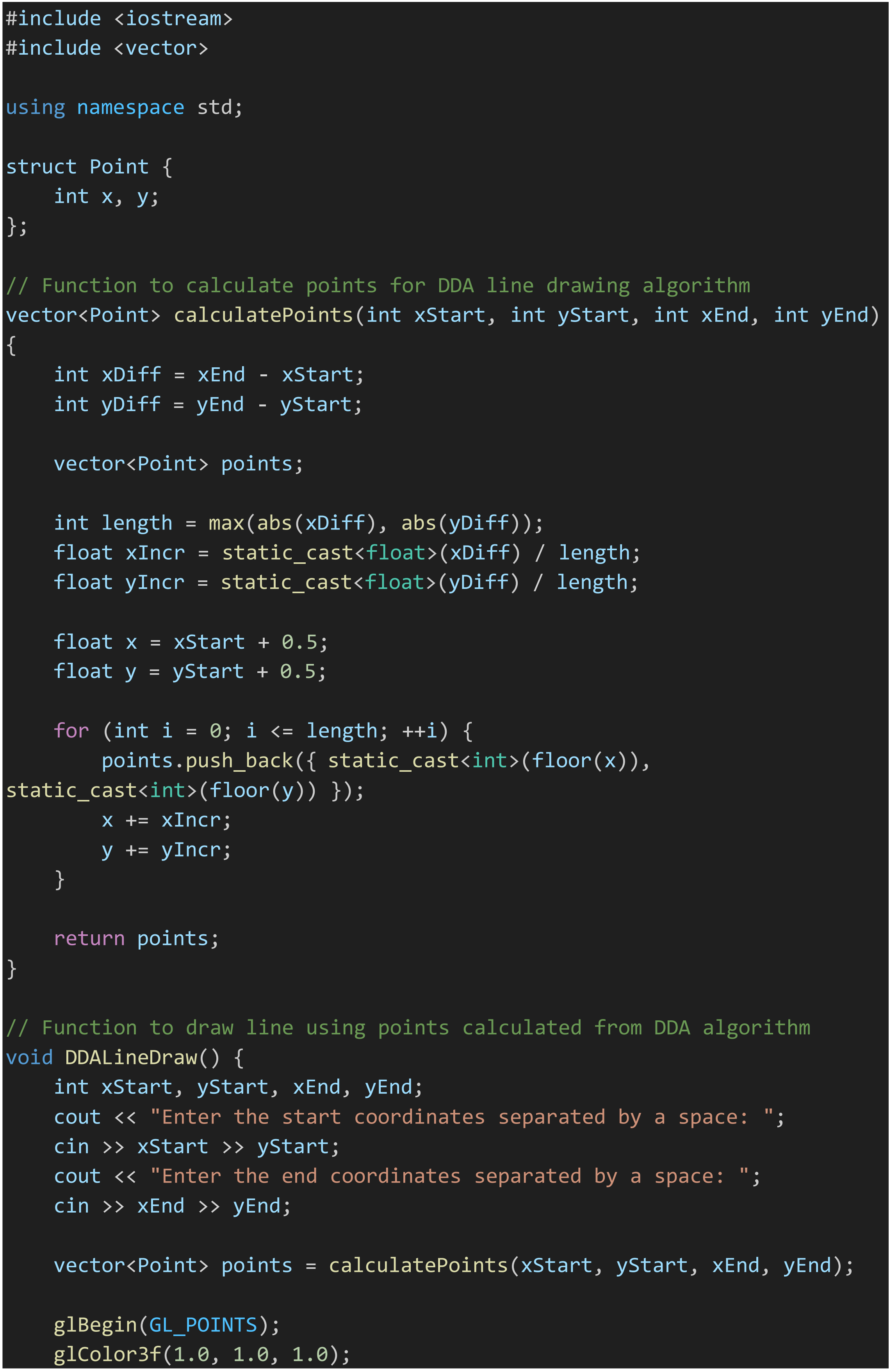
5 out of 5

Implementation details (Code can be in C/C++/Java/Python with and without using graphics library functions):

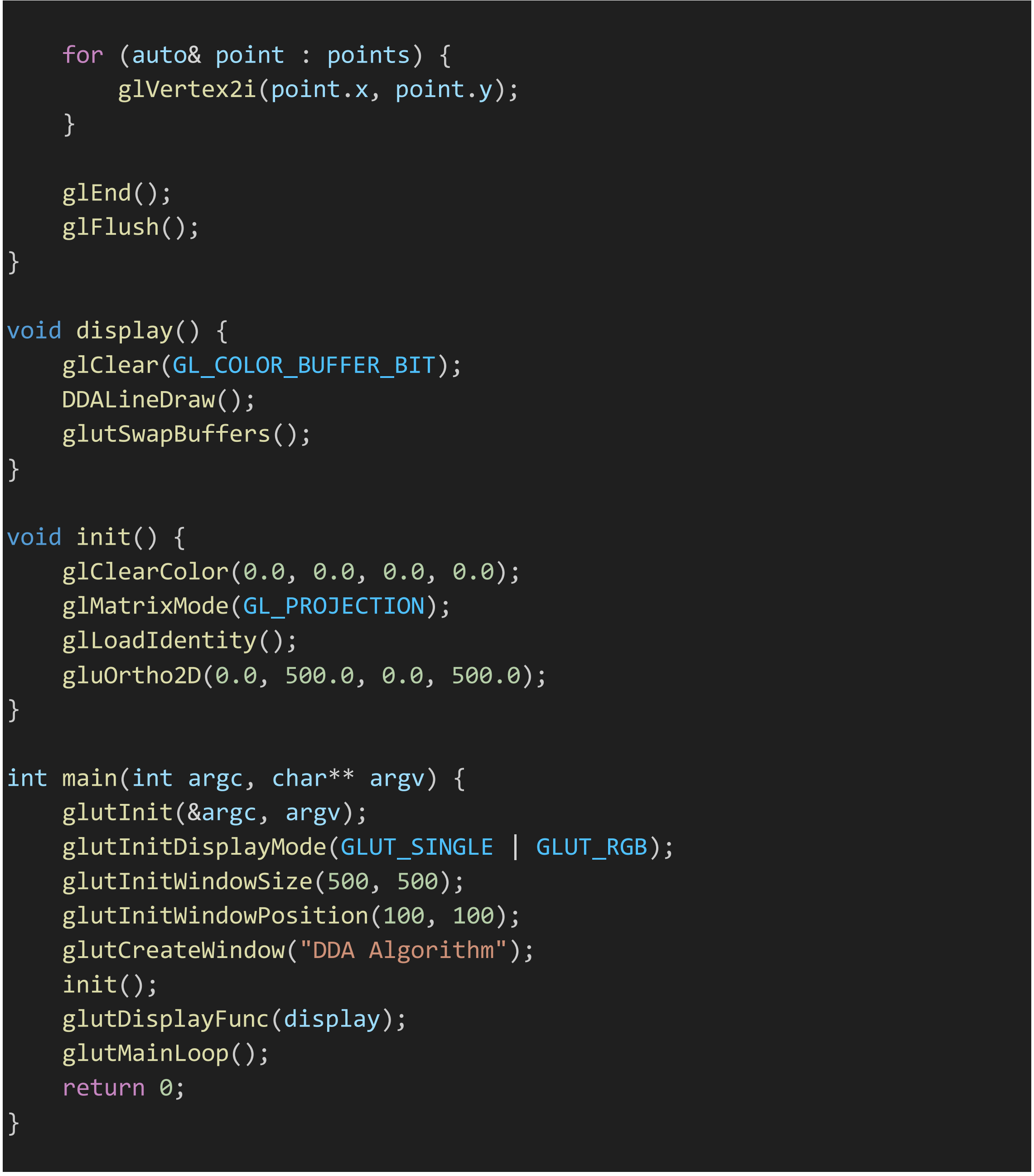
Implementation

Algorithm - DDA

```
#include <GL/glut.h>
#include <cmath>
```

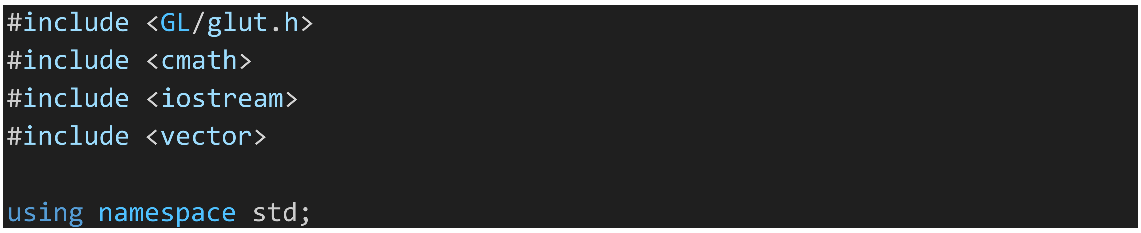


```
#include <iostream>
#include <vector>

using namespace std;

struct Point {
    int x, y;
};

// Function to calculate points for DDA line drawing algorithm
vector<Point> calculatePoints(int xStart, int yStart, int xEnd, int yEnd)
{
    int xDiff = xEnd - xStart;
    int yDiff = yEnd - yStart;

    vector<Point> points;

    int length = max(abs(xDiff), abs(yDiff));
    float xIncr = static_cast<float>(xDiff) / length;
    float yIncr = static_cast<float>(yDiff) / length;

    float x = xStart + 0.5;
    float y = yStart + 0.5;

    for (int i = 0; i <= length; ++i) {
        points.push_back({ static_cast<int>(floor(x)),
static_cast<int>(floor(y)) });
        x += xIncr;
        y += yIncr;
    }

    return points;
}

// Function to draw line using points calculated from DDA algorithm
void DDALineDraw() {
    int xStart, yStart, xEnd, yEnd;
    cout << "Enter the start coordinates separated by a space: ";
    cin >> xStart >> yStart;
    cout << "Enter the end coordinates separated by a space: ";
    cin >> xEnd >> yEnd;

    vector<Point> points = calculatePoints(xStart, yStart, xEnd, yEnd);

    glBegin(GL_POINTS);
    glColor3f(1.0, 1.0, 1.0);
```

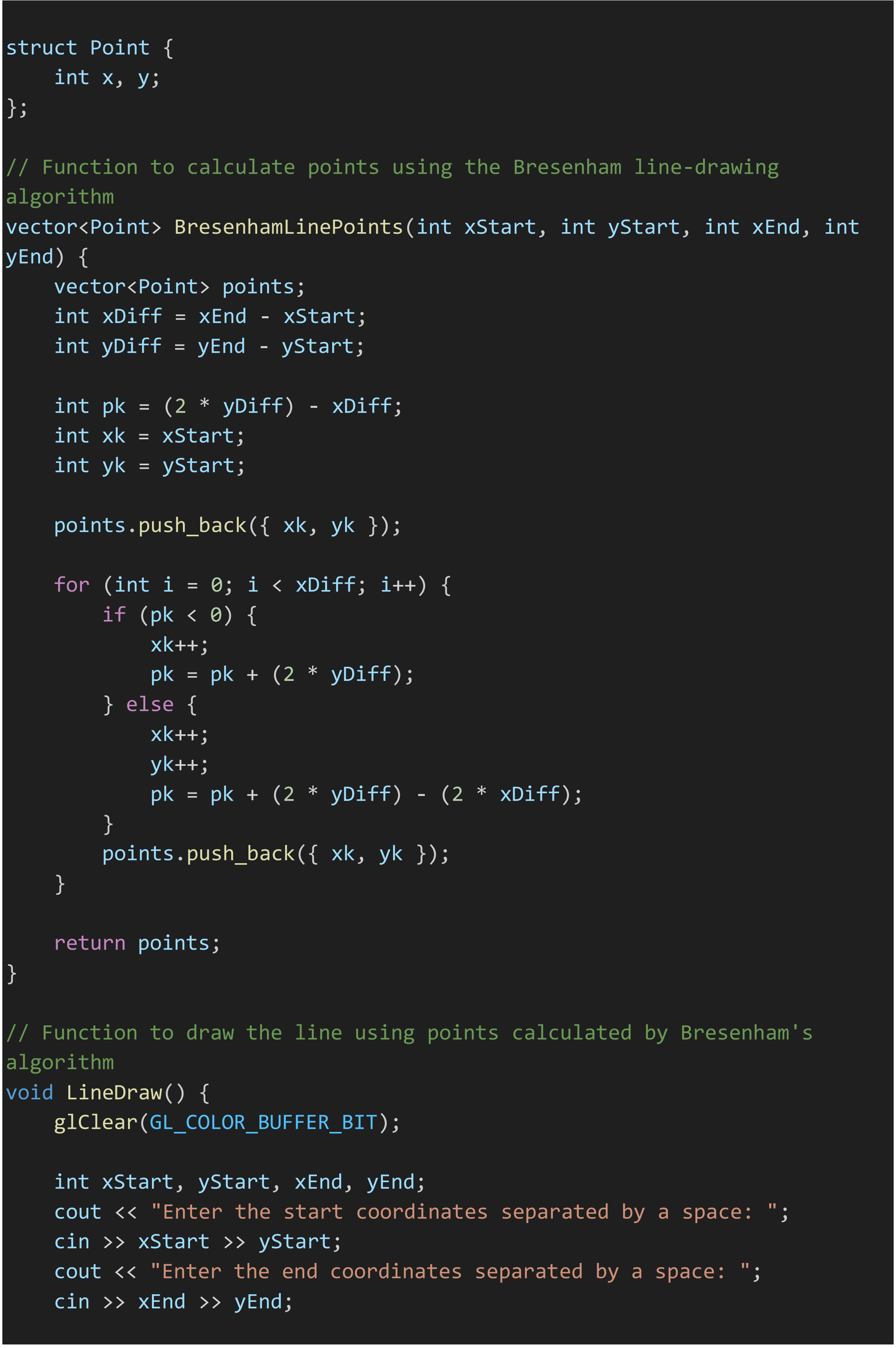


```
    for (auto& point : points) {
        glVertex2i(point.x, point.y);
    }

    glEnd();
    glFlush();
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    DDALineDraw();
    glutSwapBuffers();
}

void init() {
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 500.0, 0.0, 500.0);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("DDA Algorithm");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

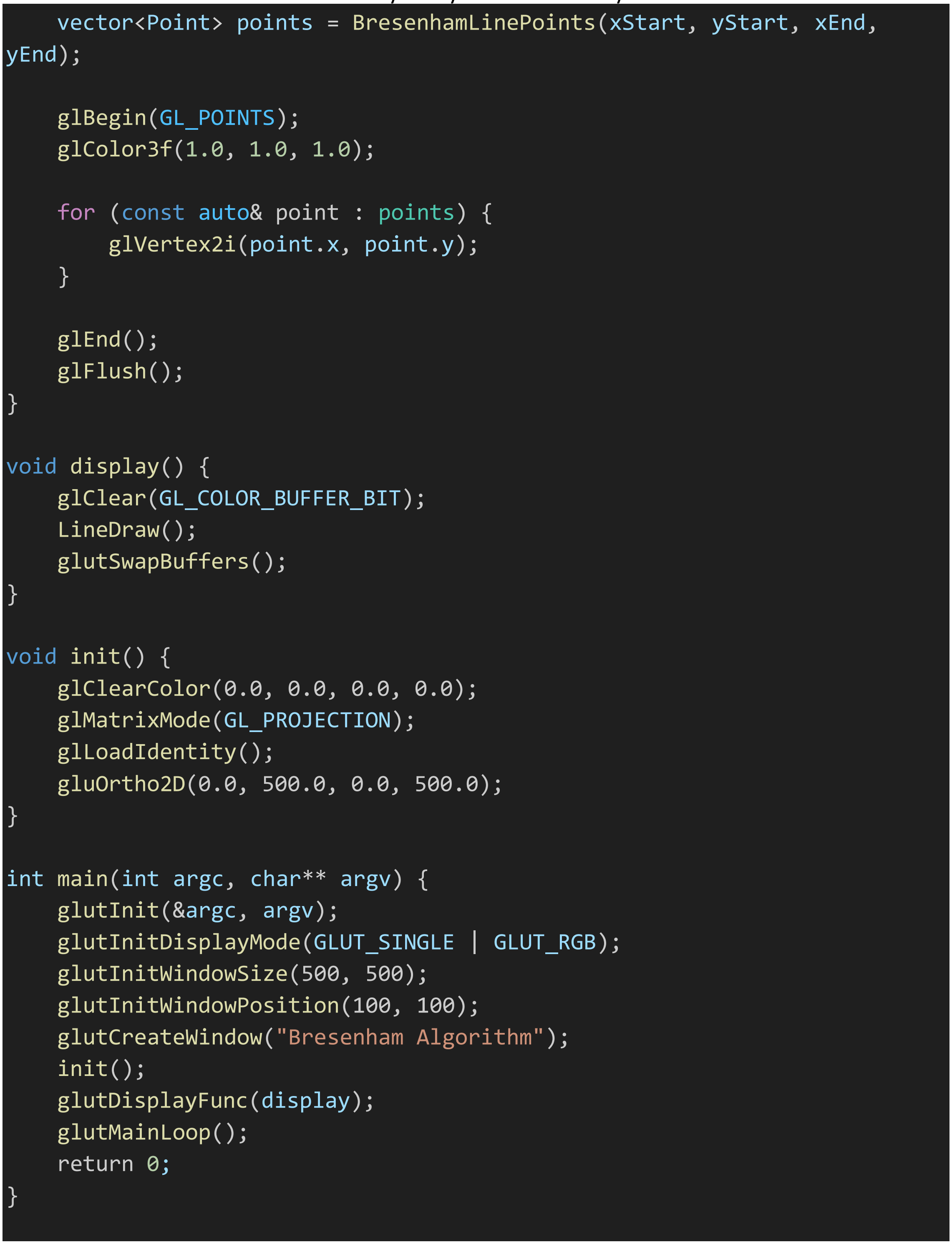
Algorithm – Bresenham's Line Drawing



```
#include <GL/glut.h>
#include <cmath>
#include <iostream>
#include <vector>

using namespace std;
```



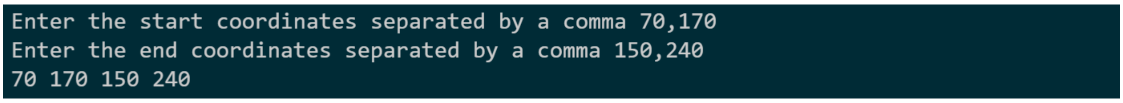
```
struct Point {
    int x, y;
};

// Function to calculate points using the Bresenham line-drawing
algorithm
vector<Point> BresenhamLinePoints(int xStart, int yStart, int xEnd, int
yEnd) {
    vector<Point> points;
    int xDiff = xEnd - xStart;
    int yDiff = yEnd - yStart;

    int pk = (2 * yDiff) - xDiff;
    int xk = xStart;
    int yk = yStart;

    points.push_back({ xk, yk });

    for (int i = 0; i < xDiff; i++) {
        if (pk < 0) {
            xk++;
            pk = pk + (2 * yDiff);
        } else {
            xk++;
            yk++;
            pk = pk + (2 * yDiff) - (2 * xDiff);
        }
        points.push_back({ xk, yk });
    }

    return points;
}

// Function to draw the line using points calculated by Bresenham's
algorithm
void LineDraw() {
    glClear(GL_COLOR_BUFFER_BIT);

    int xStart, yStart, xEnd, yEnd;
    cout << "Enter the start coordinates separated by a space: ";
    cin >> xStart >> yStart;
    cout << "Enter the end coordinates separated by a space: ";
    cin >> xEnd >> yEnd;
```



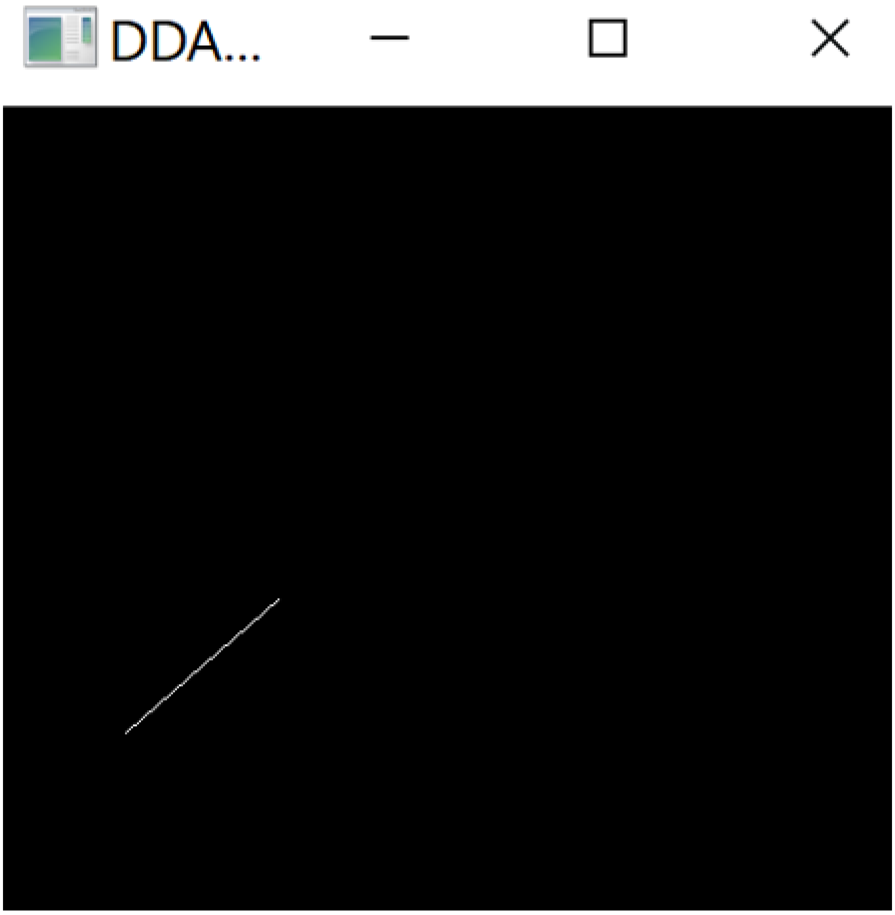
```
    vector<Point> points = BresenhamLinePoints(xStart, yStart, xEnd,
yEnd);

    glBegin(GL_POINTS);
    glColor3f(1.0, 1.0, 1.0);

    for (const auto& point : points) {
        glVertex2i(point.x, point.y);
    }

    glEnd();
    glFlush();
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    LineDraw();
    glutSwapBuffers();
}

void init() {
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 500.0, 0.0, 500.0);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Bresenham Algorithm");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Output(s) (final edited screen shot):

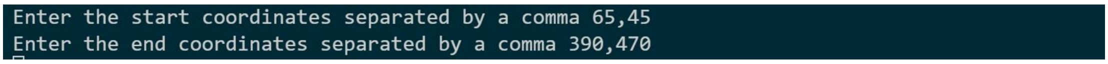
Algorithm – DDA



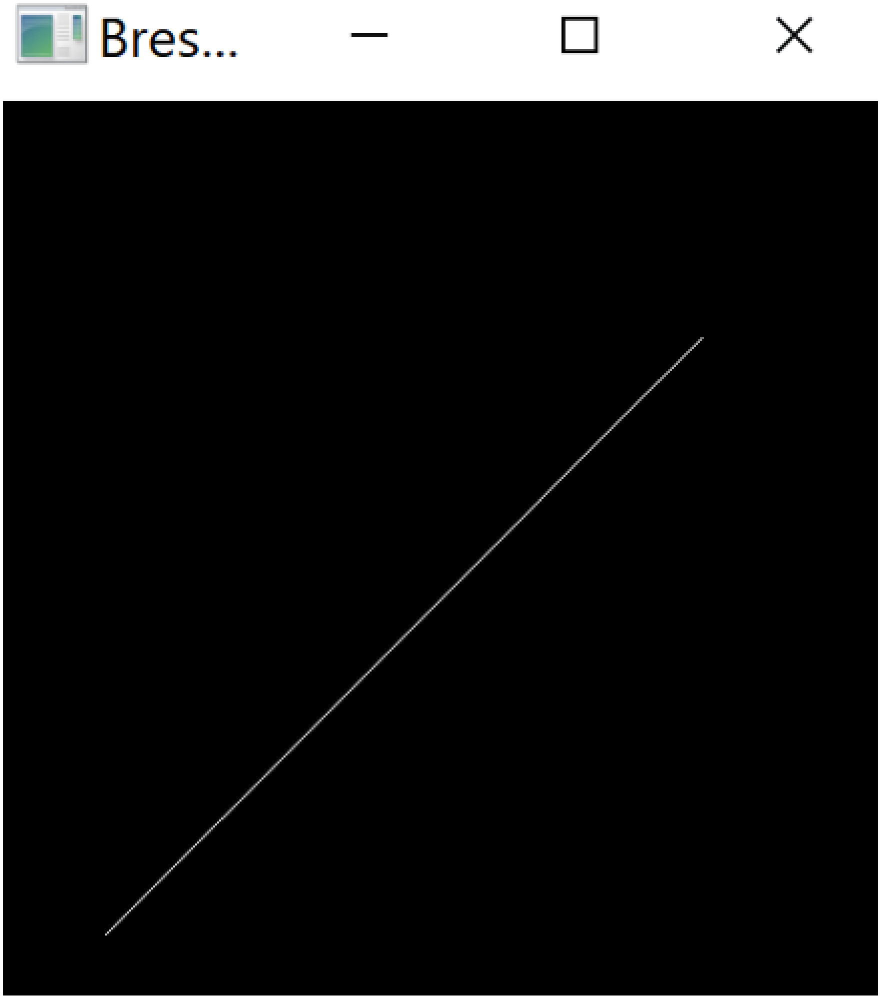
```
Enter the start coordinates separated by a comma 70,170
Enter the end coordinates separated by a comma 150,240
70 170 150 240
```



Algorithm – Bresenham’s Line Drawing



```
Enter the start coordinates separated by a comma 65,45
Enter the end coordinates separated by a comma 390,470
```



Conclusion and discussion (Comparative - compulsory):

Successfully understood and implemented both DDA and Bresenham’s Line Drawing Algorithm.

Date:

Signature of faculty in-charge

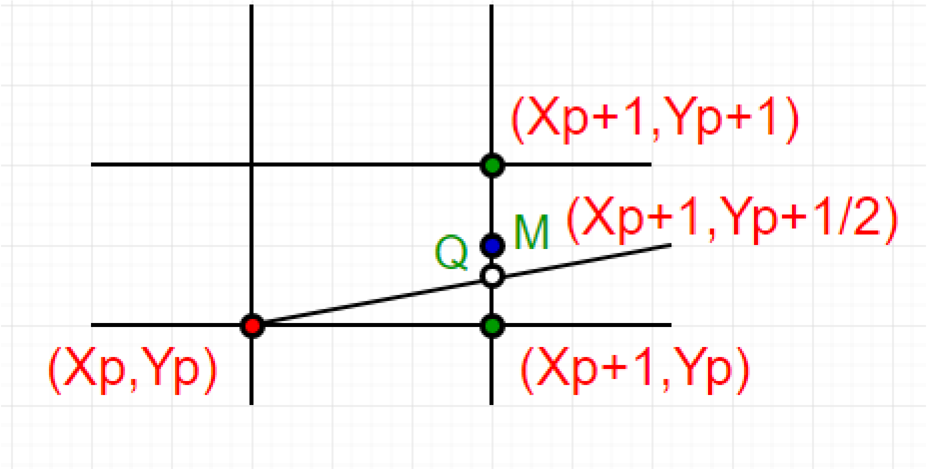
Explain Mid-point line drawing algorithm and implement it

The Mid-Point Line Drawing Algorithm is a popular algorithm used in computer graphics to draw a line between two points on a grid or raster display. It is more efficient than other line-drawing algorithms like the Digital Differential Analyzer (DDA) because it uses only integer addition, subtraction, and bit shifting, avoiding floating-point arithmetic.

The Mid-Point Line Drawing Algorithm determines the points of an intermediate pixel that lies closest to the true line path between two endpoints (x0,y0) and (x1,y1). The basic idea is to find the mid-point between two possible pixels that can be chosen and then decide which pixel is closer to the actual line.

Principle:

1. The line is drawn from left to right, pixel by pixel.
2. For every confirmed pixel, there are two choices for the next pixel, the pixel on its right ((Xp+1, Yp) in the figure) or the pixel on its upper right ((Xp+1, Yp+1) in the figure).
3. Using the mid-point algorithm, the choice between the two is made on the basis of where the intercept of the input line lies when it cuts the line between the above points.
4. If the line cuts above the midpoint ((Xp+1, Yp+0.5) in the figure), then the upper right point is chosen and if the line cuts below the midpoint then the point to the right is chosen.



Consider any given/calculated previous pixel P(Xp,Yp), there are two candidates for the next pixel closest to the line, E(Xp+1, Yp) and NE(Xp+1, Yp+1) (E stands for East and NE stands for North-East).

In Mid-Point algorithm we do following.

- Find middle of two possible next points. Middle of E(Xp+1, Yp) and NE(Xp+1, Yp+1) is M(Xp+1, Yp+1/2).
- If M is above the line, then choose E as next point.
- If M is below the line, then choose NE as next point.

These are few assumptions we make to keep the algorithm simple:

- We draw line from left to right.
- x1 < x2 and y1< y2

- Slope of the line is between 0 and 1. We draw a line from lower left to upper right.

Cases other than above assumptions can be handled using reflection.

Let us consider a line y = mx + B.
We can re-write the equation as :
y = (dy/dx)x + B or
(dy)x + B(dx) - y(dx) = 0
Let F(x, y) = (dy)x - y(dx) + B(dx) - (1)

Let we are given two end points of a line (under above assumptions)

- For all points (x,y) on the line, the solution to F(x, y) is 0.
- For all points (x,y) above the line, F(x, y) results in a negative number.
- And for all points (x,y) below the line, F(x, y) results in a positive number.

This relationship is used to determine the relative position of M
M = (Xp+1, Yp+1/2)

So our decision parameter d is,
d = F(M) = F(Xp+1, Yp+1/2)

Find new value of d from the old value of d
For simplicity, let as write F(x, y) as ax + by + c.
Where a = dy
b = -dx
c = B*dx
We got these values from above equation (1)

Case 1: If E is chosen then for next point :
dnew = F(Xp+2, Yp+1/2) = a(Xp+2) + b(Yp+1/2) + c
dold = a(Xp+1) + b(Yp+1/2) + c

Difference (Or delta) of two distances:
DELd = dnew – dold
= a(Xp+2)- a(Xp+1)+ b(Yp+1/2)- b(Yp+1/2)+ c-c
= a(Xp) +2a – a(Xp) – a
= a
Therefore, dnew = dold + dy. (as a = dy)

Case 2: If NE is chosen then for next point :
dnew = F(Xp+2, Yp+3/2) = a(Xp+2) + b(Yp+3/2) + c
dold = a(Xp+1) + b(Yp+1/2) + c

Difference (Or delta) of two distances:
DELd = dnew -dold
= a(Xp+2)- a(Xp+1)+ b(Yp+3/2)- b(Yp+1/2)+ c-c

$= a(Xp) + 2a – a(Xp) – a + b(Yp) + 3/2b – b(Yp) -1/2b$

$= a + b$

Therefore, dnew = dold + dy – dx. (as a = dy , b = -dx)

Calculation For initial value of decision parameter d0:

$d0 = F(X1+1 , Y1+1/2)$

$\quad = a(X1 + 1) + b(Y1 + 1/2) +c$

$\quad = aX1+ bY1 + c + a + b/2$

$\quad = F(X1,Y1) + a + b/2$

$\quad = a + b/2$ (as $F(X1, Y1) = 0$)

$d0 = dy – dx/2$. (as a = dy, b = -dx)

Algorithm:

```
Input (X1,Y1) and (X2,Y2)
dy = Y2- Y1
dx = X2 - X1
// initial value of
// decision parameter d
d = dy - (dx/2)
x = X1 , y = Y1

// plot initial given point
Plot(x , y)

// iterate through value of X
while(x < X2)
    x = x+1

    // 'E' is chosen
    if (d < 0)
       d = d + dy

    // 'NE' is chosen
    else
       d = d + dy - dx
       y = y+1
    Plot(x,y)
```