CS 352

Computer Graphics & Visualization

Assignment - 3

Name – Anjani Kumar

Roll No - 210001004

Question – 1: DDA LINE DRAWING ALGORITHM

Code:

```
#include <GL/glut.h>
#include<bits/stdc++.h>
#include <iostream>
#include <math.h>
#include <vector>

using namespace std;

// Global variables to store coordinates
float x1_coor, y1_coor, x2_coor, y2_coor;

// Vectors to store calculated coordinates
vector<float> x_coor, y_coor;

// Function to handle lines with slope greater than one
void slope_greater_than_one(int steps) {
x_coor.push_back(x1_coor);
y_coor.push_back(y1_coor);

float curr_x = x1_coor, curr_y = y1_coor;
for (int i = 1; i <= steps; i++) {
curr_y++;
curr_x = curr_x + float((x2_coor - x1_coor) / (y2_coor - y1_coor));
x_coor.push_back(round(curr_x));
```

```
y_coor.push_back(curr_y);
}

// Plot the calculated points
for (int i = 0; i < x_coor.size(); i++) {
glBegin(GL_POINTS);
glVertex2f(x_coor[i], y_coor[i]);
glEnd();
}
}

// Function to handle lines with slope less than one
void slope_less_than_one(int steps) {
x_coor.push_back(x1_coor);
y_coor.push_back(y1_coor);
//storing the points
float curr_x = x1_coor, curr_y = y1_coor;
for (int i = 1; i <= steps; i++) {
curr_x++;
curr_y = curr_y + float((y2_coor - y1_coor) / (x2_coor - x1_coor));
x_coor.push_back(round(curr_x));
y_coor.push_back(round(curr_y));
}

// Plot the calculated points
for (int i = 0; i < x_coor.size(); i++) {
glBegin(GL_POINTS);
glVertex2f(x_coor[i], y_coor[i]);
glEnd();
}
}

// Function to handle lines with slope equal to one
```

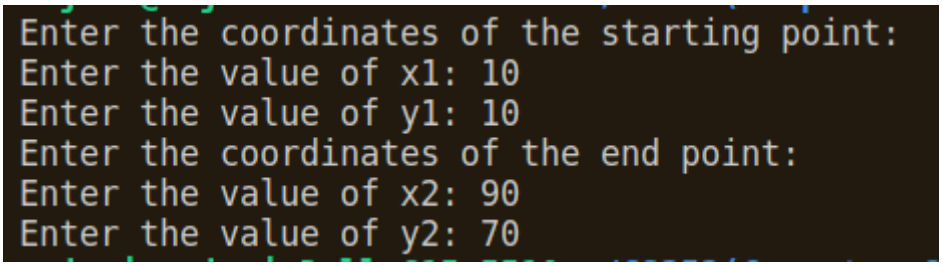
```
void slope_equal_to_one(int steps) {
x_coor.push_back(x1_coor);
y_coor.push_back(y1_coor);

float curr_x = x1_coor, curr_y = y1_coor;
for (int i = 1; i <= steps; i++) {
curr_y++;
curr_x++;
x_coor.push_back(curr_x);
y_coor.push_back(curr_y);
}

// Plot the calculated points
for (int i = 0; i < x_coor.size(); i++) {
glBegin(GL_POINTS);
glVertex2f(x_coor[i], y_coor[i]);
glEnd();
}
}

// Display callback function
void displayCB() {
// Ensure the starting point is on the left
if (x1_coor > x2_coor) {
swap(x1_coor, x2_coor);
swap(y1_coor, y2_coor);
}

// Calculate the differences in x and y coordinates
int dx = x2_coor - x1_coor;
int dy = y2_coor - y1_coor;

// Determine the number of steps
```

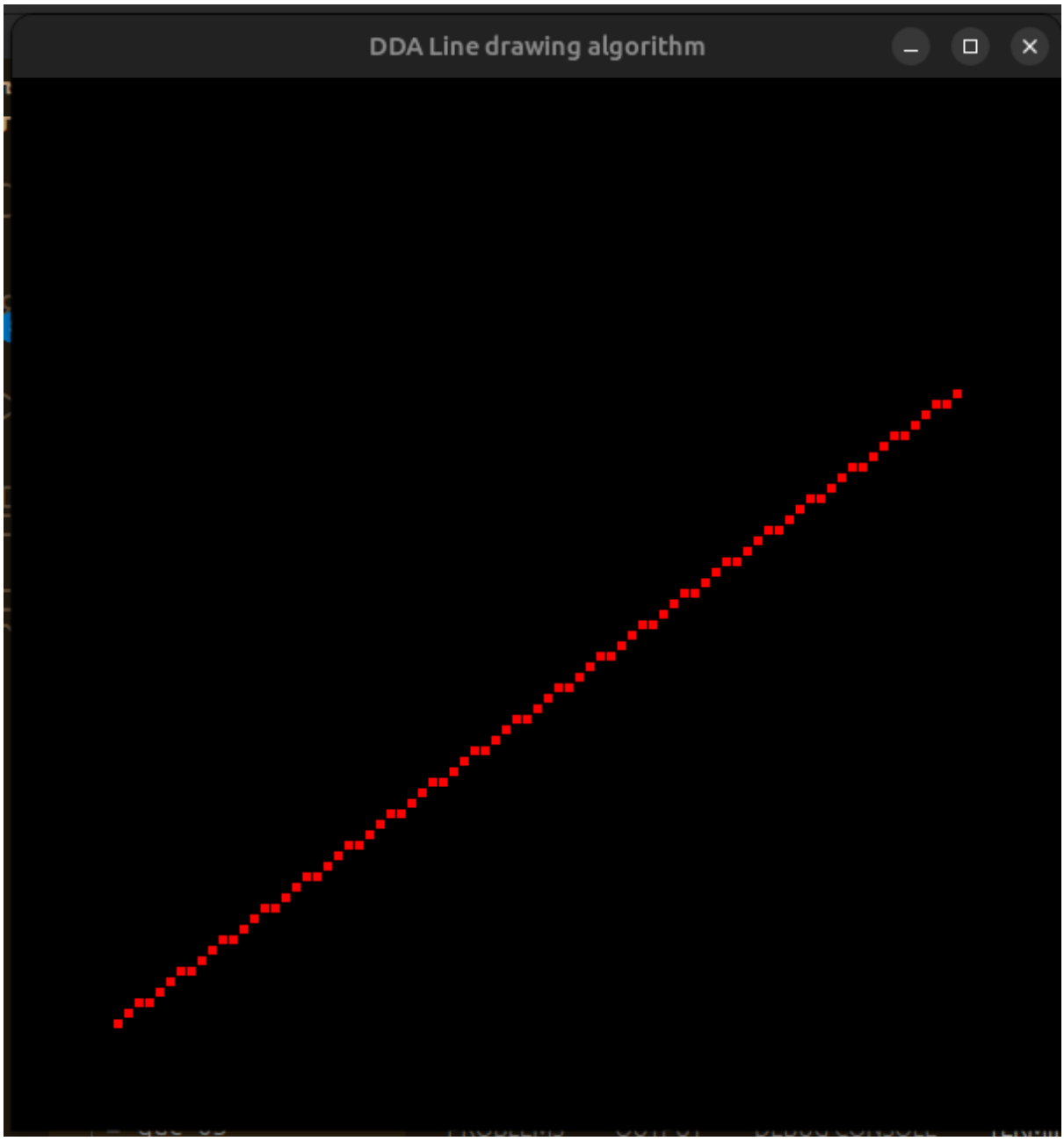
```
int steps = max(abs(dx), abs(dy));

// Determine the slope type and call the appropriate function
if (abs(dy) > abs(dx)) {
// Slope greater than one
slope_greater_than_one(steps);
} else if (abs(dy) < abs(dx)) {
// Slope less than one
slope_less_than_one(steps);
} else {
// Slope equal to one
slope_equal_to_one(steps);
}

// Flush the drawing commands
glFlush();
}

// Main function
int main(int argc, char *argv[]) {
// Prompt user to enter coordinates
cout << "Enter the coordinates of the starting point: \n";
cout << "Enter the value of x1: ";
cin >> x1_coor;
cout << "Enter the value of y1: ";
cin >> y1_coor;

cout << "Enter the coordinates of the end point: \n";
cout << "Enter the value of x2: ";
cin >> x2_coor;
cout << "Enter the value of y2: ";
cin >> y2_coor;
```

```
// Initialize GLUT and set display mode
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGB);
glutInitWindowSize(600, 600);
glutCreateWindow("DDA Line drawing algorithm");

// Set OpenGL properties
glClearColor(1, 1, 1, 0.0);
glColor3f(1, 0, 0);
glPointSize(5.0);
gluOrtho2D(0, 100, 0, 100);

// Set display callback function
glutDisplayFunc(displayCB);

// Enter GLUT event processing loop
glutMainLoop();

return 0;
}
```

Screenshot of the outputs:

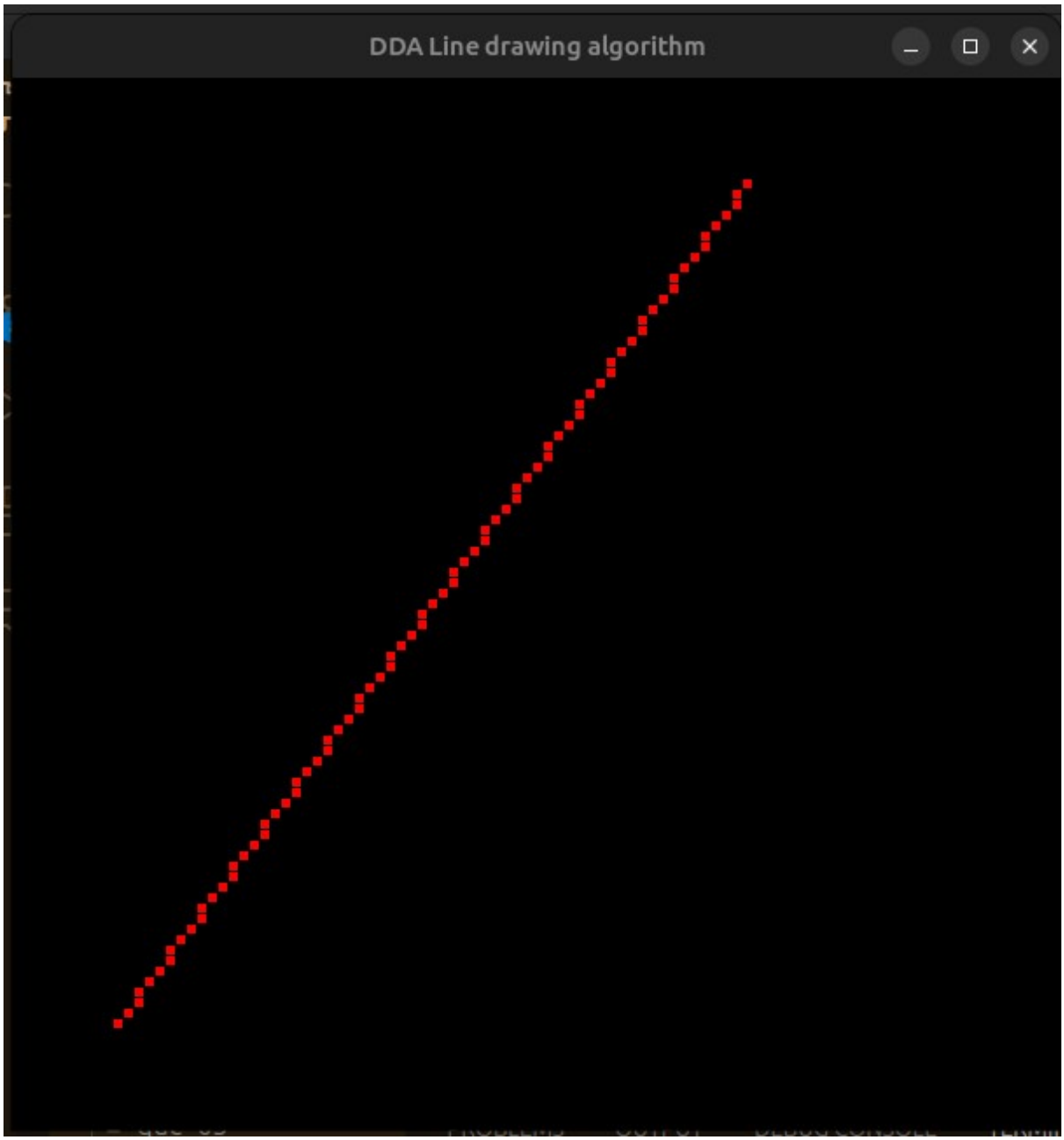
i) For slope < 1





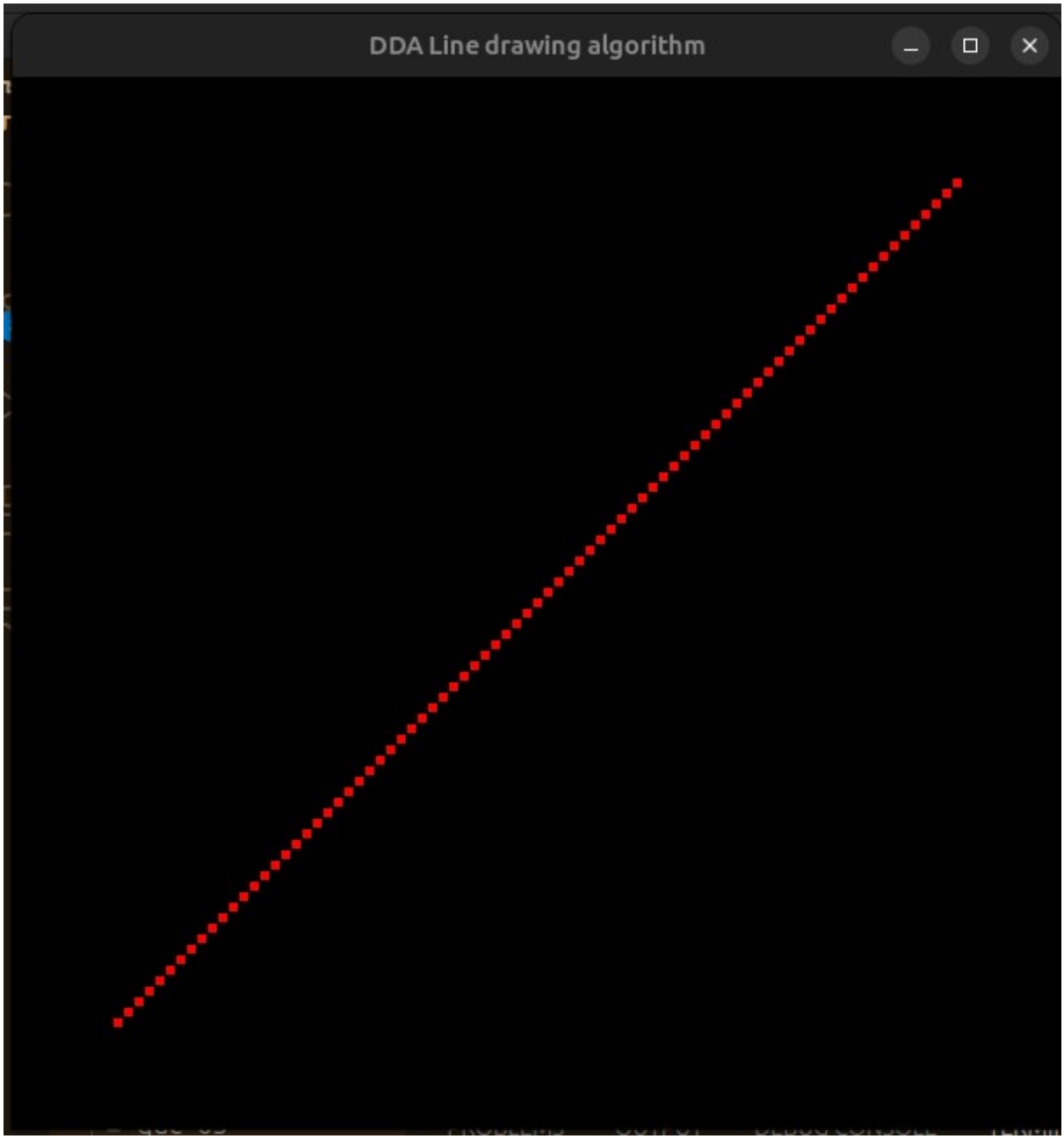
ii) For slope>1

```
Enter the coordinates of the starting point:
Enter the value of x1: 10
Enter the value of y1: 10
Enter the coordinates of the end point:
Enter the value of x2: 70
Enter the value of y2: 90
```



iii) For slope = 1

```
Enter the coordinates of the starting point:
Enter the value of x1: 90
Enter the value of y1: 90
Enter the coordinates of the end point:
Enter the value of x2: 10
Enter the value of y2: 10
```


DDA Line drawing algorithm

Question 2: Bresenham's Line Drawing Algorithm

Code:

```
#include <GL/glut.h>
#include<bits/stdc++.h>
using namespace std;
// Global variables to store coordinates
float x1_coor, y1_coor, x2_coor, y2_coor;
// Vectors to store calculated coordinates
vector<float> x_coor, y_coor;
// Function to handle lines with slope greater than one
void slope_greater_than_one(int steps) {
x_coor.push_back(x1_coor);
y_coor.push_back(y1_coor);
int dy = y2_coor - y1_coor;
int dx = x2_coor - x1_coor;
int decision_parameter = 2*dx - dy;
int curr_x = x1_coor;
int curr_y = y1_coor;
while(curr_y!=y2_coor){
curr_y++;
if(decision_parameter<0){
decision_parameter += 2*dx;
}
else {
decision_parameter += 2*(dx-dy);
curr_x++;
}
x_coor.push_back(curr_x);
y_coor.push_back(curr_y);
```

```
}

for(int i=0;i<x_coor.size();i++){
glBegin(GL_POINTS);
glVertex2f(x_coor[i],y_coor[i]);
glEnd();
}
}

// Function to handle lines with slope less than one
void slope_less_than_one() {
x_coor.push_back(x1_coor);
y_coor.push_back(y1_coor);
int dy = y2_coor - y1_coor;
int dx = x2_coor - x1_coor;
int decision_parameter = 2*dy - dx;
int curr_x = x1_coor;
int curr_y = y1_coor;
while(curr_x!=x2_coor){
curr_x++;
if(decision_parameter<0){
decision_parameter += 2*dy;
}
else {
decision_parameter += 2*(dy-dx);
curr_y++;
}
x_coor.push_back(curr_x);
y_coor.push_back(curr_y);
}
```

```
for(int i=0;i<x_coor.size();i++){
glBegin(GL_POINTS);
glVertex2f(x_coor[i],y_coor[i]);
glEnd();
}
}
// Function to handle lines with slope equal to one
void slope_equal_to_one(int steps) {
x_coor.push_back(x1_coor);
y_coor.push_back(y1_coor);
int dy = y2_coor - y1_coor;
int dx = x2_coor - x1_coor;
int decision_parameter = 2*dy - dx;
int curr_x = x1_coor;
int curr_y = y1_coor;
while(curr_x!=x2_coor){
curr_x++;
if(decision_parameter<0){
decision_parameter += 2*dy;
}
else {
decision_parameter += 2*(dy-dx);
curr_y++;
}
x_coor.push_back(curr_x);
y_coor.push_back(curr_y);
}
for(int i=0;i<x_coor.size();i++){
glBegin(GL_POINTS);
glVertex2f(x_coor[i],y_coor[i]);
glEnd();
```

```
}
}

// Display callback function
void displayCB() {
// Ensure the starting point is on the left
if (x1_coor > x2_coor) {
swap(x1_coor, x2_coor);
swap(y1_coor, y2_coor);
}

// Calculate the differences in x and y coordinates
int dx = x2_coor - x1_coor;
int dy = y2_coor - y1_coor;

// Determine the number of steps
int steps = max(abs(dx), abs(dy));

// Determine the slope type and call the appropriate function
if (abs(dy) > abs(dx)) {
// Slope greater than one
slope_greater_than_one(steps);
} else if (abs(dy) < abs(dx)) {
// Slope less than one
slope_less_than_one();
} else {
// Slope equal to one
slope_equal_to_one(steps);
}
// Flush the drawing commands
glFlush();
```

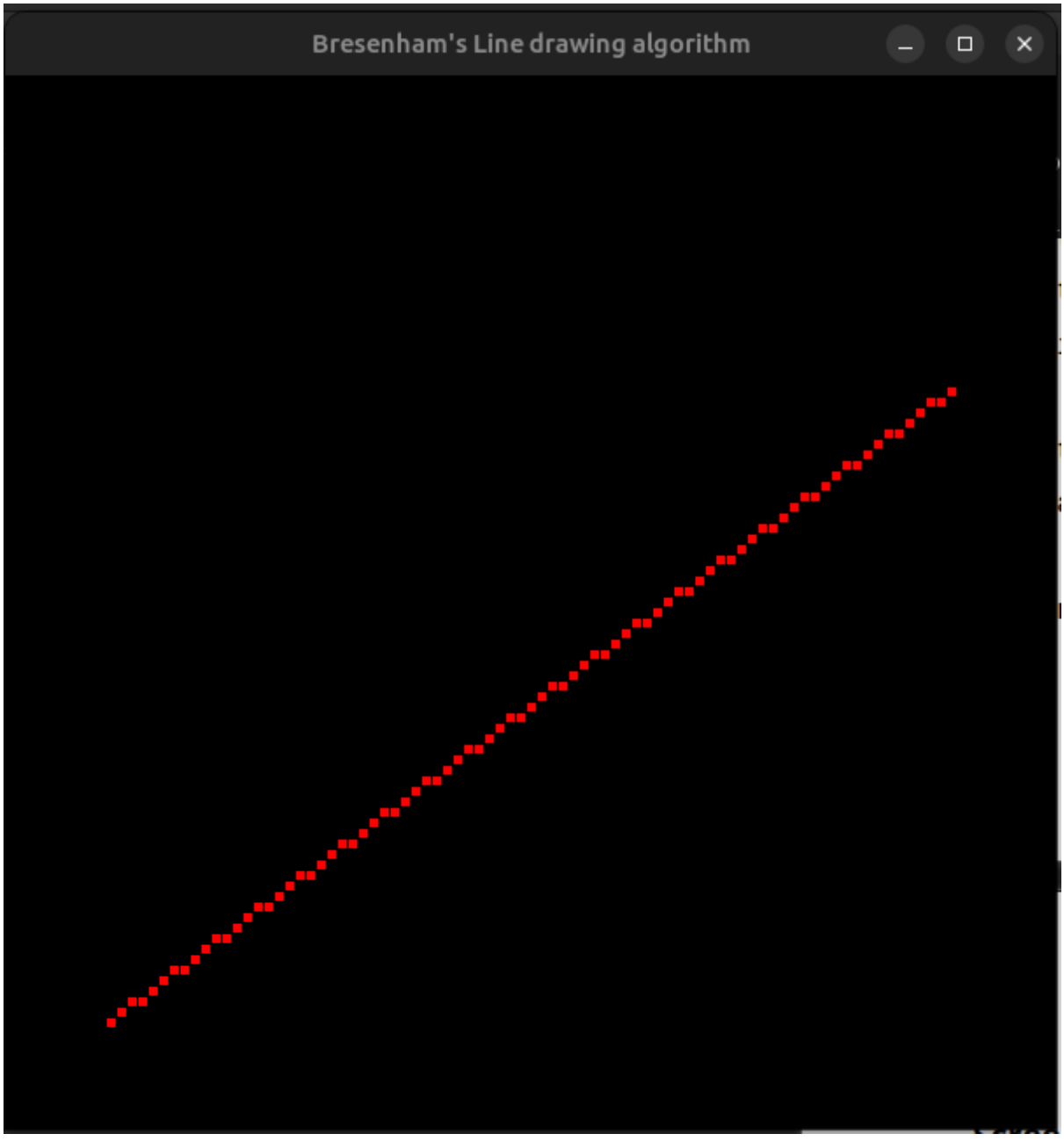
```
}
// Main function
int main(int argc, char *argv[]) {
// Prompt user to enter coordinates
cout << "Enter the coordinates of the starting point: \n";
cout << "Enter the value of x1: ";
cin >> x1_coor;
cout << "Enter the value of y1: ";
cin >> y1_coor;
cout << "Enter the coordinates of the end point: \n";
cout << "Enter the value of x2: ";
cin >> x2_coor;
cout << "Enter the value of y2: ";
cin >> y2_coor;
// Initialize GLUT and set display mode
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGB);
glutInitWindowSize(600, 600);
glutCreateWindow("Bresenham's Line drawing algorithm");
// Set OpenGL properties
glClearColor(1, 1, 1, 0.0);
glColor3f(1, 0, 0);
glPointSize(5.0);
gluOrtho2D(0, 100, 0, 100);
// Set display callback function
glutDisplayFunc(displayCB);

// Enter GLUT event processing loop
glutMainLoop();

return 0;
```

```
}
```

Screenshot of the Output:

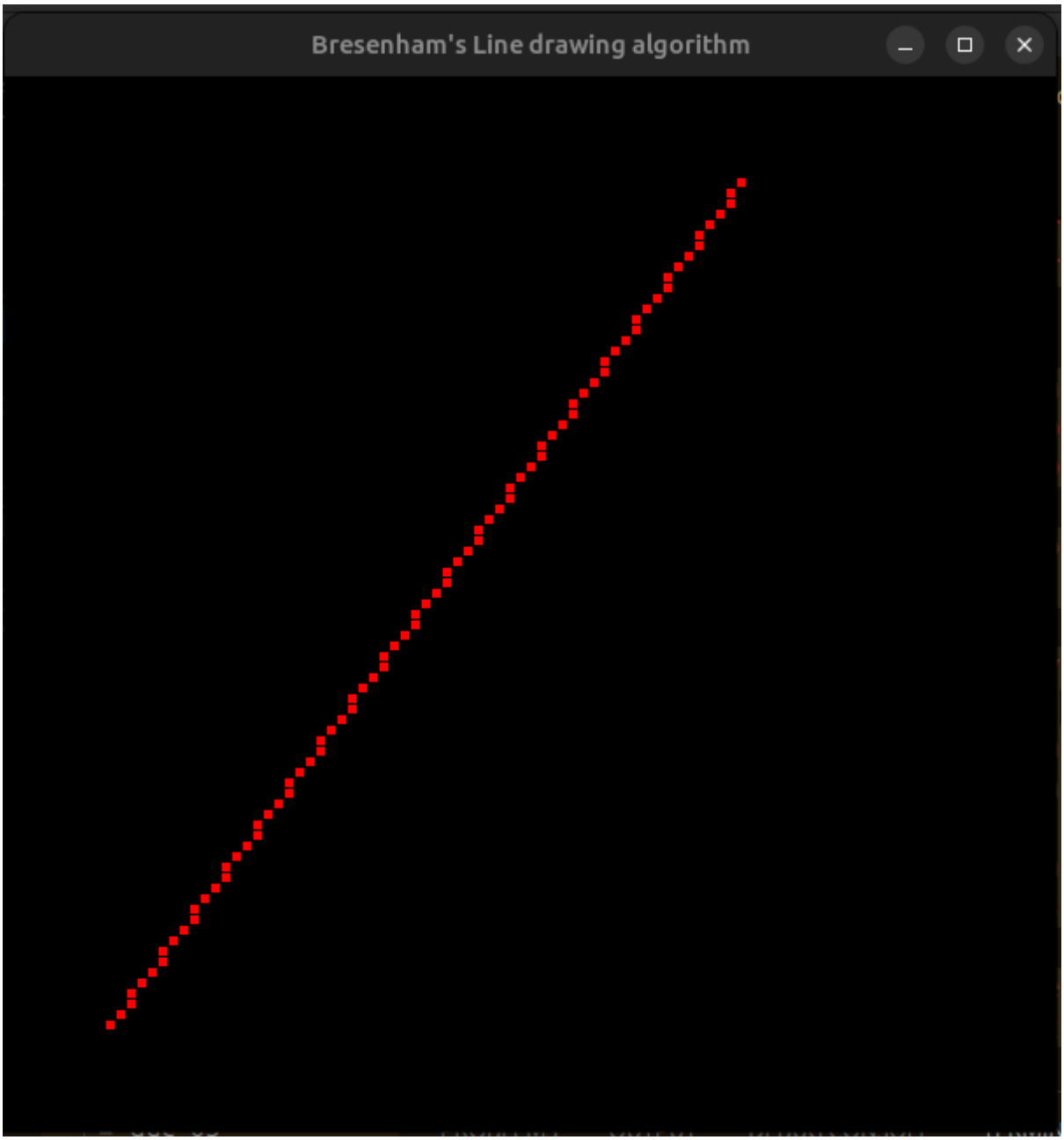
i) For slope<1

```
Enter the coordinates of the starting point:
Enter the value of x1: 10
Enter the value of y1: 10
Enter the coordinates of the end point:
Enter the value of x2: 90
Enter the value of y2: 70
```



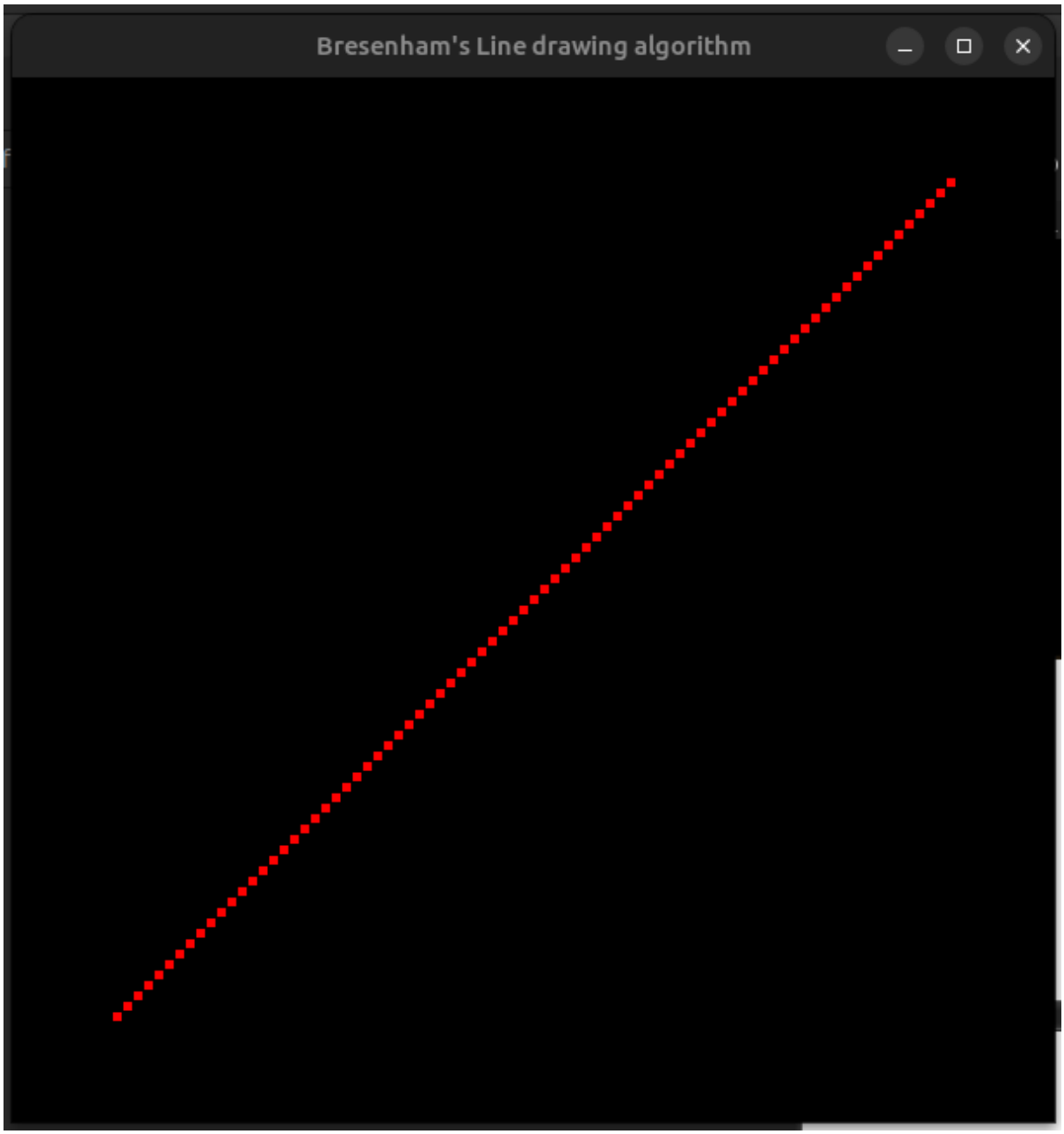
ii) For slope>1

```
Enter the coordinates of the starting point:
Enter the value of x1: 10
Enter the value of y1: 10
Enter the coordinates of the end point:
Enter the value of x2: 70
Enter the value of y2: 90
```



iii) For slope =1

```
Enter the coordinates of the starting point:
Enter the value of x1: 10
Enter the value of y1: 10
Enter the coordinates of the end point:
Enter the value of x2: 90
Enter the value of y2: 90
```



Question 3: Mid point Line Drawing Algorithm

Code:

```
#include <GL/glut.h>
#include<bits/stdc++.h>
using namespace std;
// Global variables to store coordinates
float x1_coor, y1_coor, x2_coor, y2_coor;
// Vectors to store calculated coordinates
vector<float> x_coor, y_coor;
// Function to handle lines with slope greater than one
void slope_greater_than_one(int steps) {
x_coor.push_back(x1_coor);
y_coor.push_back(y1_coor);
int dy = y2_coor - y1_coor;
int dx = x2_coor - x1_coor;
int decision_parameter = dx - dy/2;
int curr_x = x1_coor;
int curr_y = y1_coor;
while(curr_y!=y2_coor){
curr_y++;
if(decision_parameter<0){
decision_parameter += dx;
}
else {
decision_parameter += (dx-dy);
curr_x++;
}
x_coor.push_back(curr_x);
y_coor.push_back(curr_y);
}
```

```
for(int i=0;i<x_coor.size();i++){
glBegin(GL_POINTS);
glVertex2f(x_coor[i],y_coor[i]);
glEnd();
}
}
// Function to handle lines with slope less than one
void slope_less_than_one() {
x_coor.push_back(x1_coor);
y_coor.push_back(y1_coor);
int dy = y2_coor - y1_coor;
int dx = x2_coor - x1_coor;
int decision_parameter = dy - dx/2;
int curr_x = x1_coor;
int curr_y = y1_coor;
while(curr_x!=x2_coor){
curr_x++;
if(decision_parameter<0){
decision_parameter += dy;
}
else {
decision_parameter += (dy-dx);
curr_y++;
}
x_coor.push_back(curr_x);
y_coor.push_back(curr_y);
}
for(int i=0;i<x_coor.size();i++){
glBegin(GL_POINTS);
glVertex2f(x_coor[i],y_coor[i]);
```

```
glEnd();
}
}
// Function to handle lines with slope equal to one
void slope_equal_to_one(int steps) {
x_coor.push_back(x1_coor);
y_coor.push_back(y1_coor);
int dy = y2_coor - y1_coor;
int dx = x2_coor - x1_coor;
int decision_parameter = 2*dy - dx;
int curr_x = x1_coor;
int curr_y = y1_coor;
while(curr_x!=x2_coor){
curr_x++;
if(decision_parameter<0){
decision_parameter += 2*dy;
}
else {
decision_parameter += 2*(dy-dx);
curr_y++;
}
x_coor.push_back(curr_x);
y_coor.push_back(curr_y);
}
for(int i=0;i<x_coor.size();i++){
glBegin(GL_POINTS);
glVertex2f(x_coor[i],y_coor[i]);
glEnd();
}
}
```

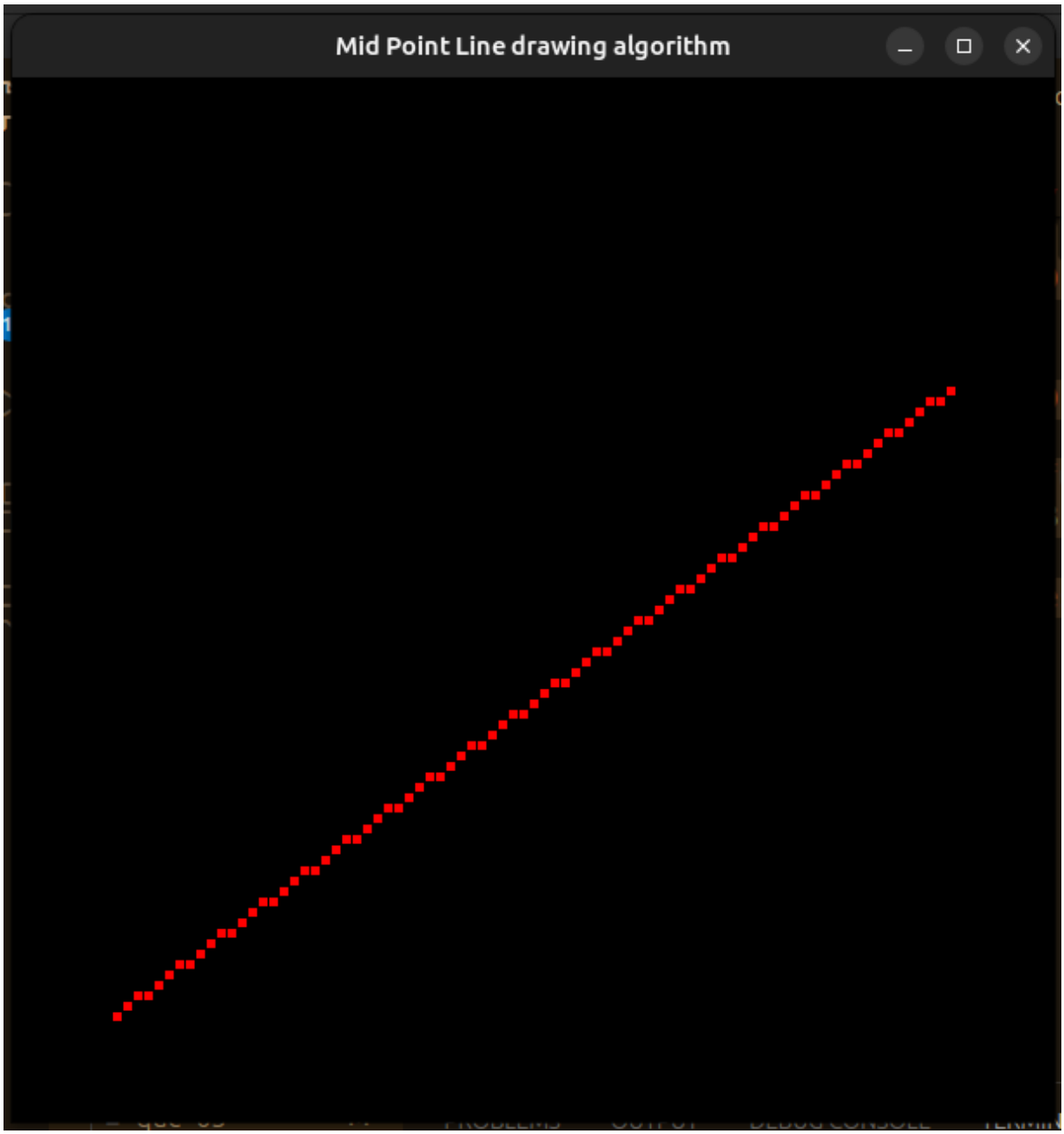
```
// Display callback function
void displayCB() {
// Ensure the starting point is on the left
if (x1_coor > x2_coor) {
swap(x1_coor, x2_coor);
swap(y1_coor, y2_coor);
}
// Calculate the differences in x and y coordinates
int dx = x2_coor - x1_coor;
int dy = y2_coor - y1_coor;
// Determine the number of steps
int steps = max(abs(dx), abs(dy));
// Determine the slope type and call the appropriate function
if (abs(dy) > abs(dx)) {
// Slope greater than one
slope_greater_than_one(steps);
} else if (abs(dy) < abs(dx)) {
// Slope less than one
slope_less_than_one();
} else {
// Slope equal to one
slope_equal_to_one(steps);
}
// Flush the drawing commands
glFlush();
}
// Main function
int main(int argc, char *argv[]) {
// Prompt user to enter coordinates
cout << "Enter the coordinates of the starting point: \n";
cout << "Enter the value of x1: ";
```

```
cin >> x1_coor;
cout << "Enter the value of y1: ";
cin >> y1_coor;
cout << "Enter the coordinates of the end point: \n";
cout << "Enter the value of x2: ";
cin >> x2_coor;
cout << "Enter the value of y2: ";
cin >> y2_coor;
// Initialize GLUT and set display mode
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGB);
glutInitWindowSize(600, 600);
glutCreateWindow("Mid Point Line drawing algorithm");
// Set OpenGL properties
glClearColor(1, 1, 1, 0.0);
glColor3f(1, 0, 0);
glPointSize(5.0);
gluOrtho2D(0, 100, 0, 100);

// Set display callback function
glutDisplayFunc(displayCB);

// Enter GLUT event processing loop
glutMainLoop();

return 0;
}
```

Screenshot of the outputs:

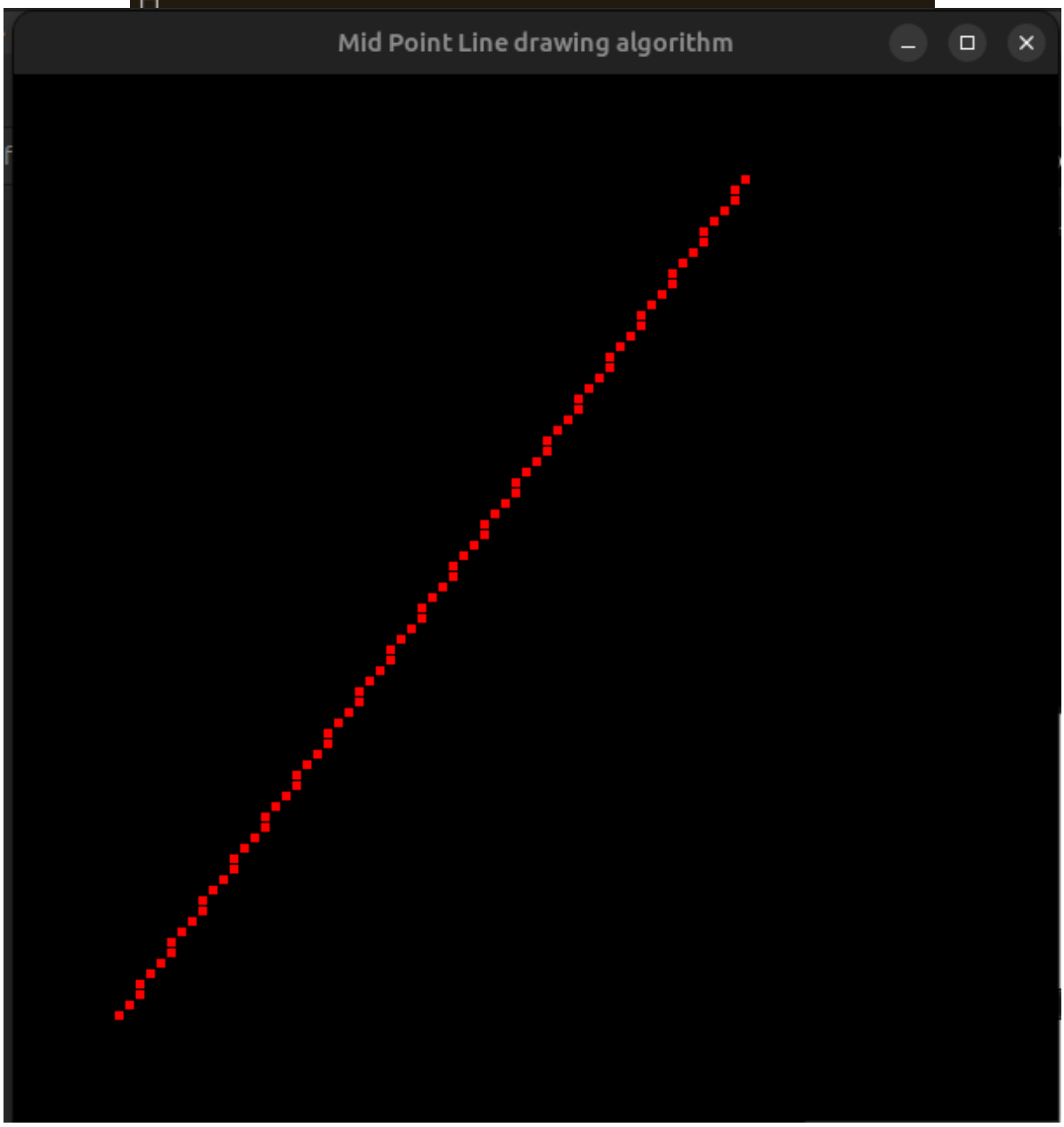
i) Slope<1

```
Enter the coordinates of the starting point:
Enter the value of x1: 10
Enter the value of y1: 10
Enter the coordinates of the end point:
Enter the value of x2: 90
Enter the value of y2: 70
```



ii) Slope>1

```
Enter the coordinates of the starting point:
Enter the value of x1: 10
Enter the value of y1: 10
Enter the coordinates of the end point:
Enter the value of x2: 70
Enter the value of y2: 90
```



iii) slope=1

```
Enter the coordinates of the starting point:
Enter the value of x1: 10
Enter the value of y1: 10
Enter the coordinates of the end point:
Enter the value of x2: 90
Enter the value of y2: 90
```