Jahangirnagar University

3rd Year 1st Semester Examination 2022

***Course title**: Computer Graphics Laboratory*

***Course code**: CSE-304*

Lab Report -6

Submitted to-

Dr. Mohammad Shorif Uddin

Professor

Department of Computer Science and Engineering

Jahangirnagar University

Dr. Morium Akter

Associate Professor

Department of Computer Science and Engineering

Jahangirnagar University

Submitted by:

Name : Akila Nipo

Class Roll : 368

Exam Roll : 202180

❖ The Liang-Barsky Algorithm:

The Liang-Barsky algorithm categorizes the line segments into three types: those that are completely outside the clipping window, those that are completely inside, and those that partially overlap with the window. It uses parametric equations to determine the intersections of the line with the clipping window boundaries and calculates the appropriate u-values (parametric coordinates) that define the visible portion of the line.

$$\begin{cases} x = x_1 + \Delta x \cdot u \\ y = y_1 + \Delta y \cdot u \end{cases}$$

Now consider the tools we need to turn this basic idea into an efficient algorithm. For point (x, y) inside the clipping window, we have

$$x_{\min} \le x_1 + \Delta x \cdot u \le x_{\max}$$
$$y_{\min} \le y_1 + \Delta y \cdot u \le y_{\max}$$

Rewrite the four inequalities as

$$p_k\, u \le q_k, \qquad k = 1, 2, 3, 4$$

where

$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{\min} \qquad \text{(left)}$$
$$p_2 = \Delta x \qquad q_2 = x_{\max} - x_1 \qquad \text{(right)}$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{\min} \qquad \text{(bottom)}$$
$$p_4 = \Delta y \qquad q_4 = y_{\max} - y_1 \qquad \text{(top)}$$

❖ The Liang-Barsky Algorithm:

Let R be the rectangular window whose lower left-hand corner is at L(—3, 1) and upper right-hand corner is at R(2, 6).

Line:A(-4,2)B(-1,7)

Line:C(-1,5)D(3,8)

Line:E(-2,3)F(1,2)

Line:G(1,-2)H(3,3)

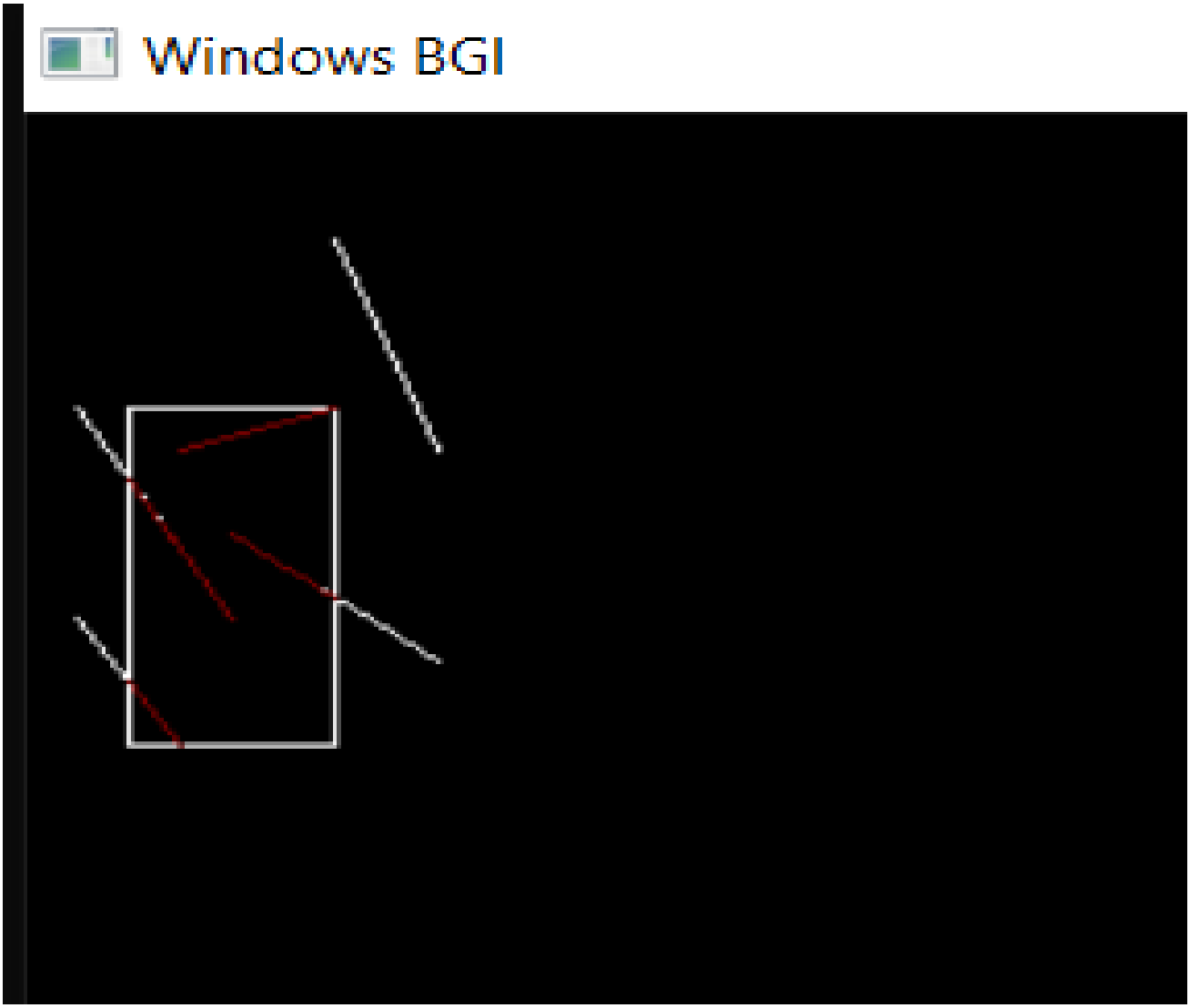
Line:I(-4,7)J(-2,10)

Source Code:

```
#include <iostream>
#include <graphics.h>
using namespace std;

struct lines {
    int x1, y1, x2, y2;
};

int xmin, xmax, ymin, ymax;

int sign(int x) {
    if (x > 0)
        return 1;
    else
        return 0;
}

void clip(struct lines mylines) {
    // ... (rest of the clip
function remains the same)
    // Modify the code to draw using
graphics.h
    setcolor(GREEN);
    line(mylines.x1, mylines.y1,
mylines.x2, mylines.y2);
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm,
"C:\\TC\\BGI");  // Set your BGI
path

    xmin = -30 + 50;
    xmax = 10 + 50;
    ymin = 20 + 50;
    ymax = 100 + 50;

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

```
struct lines mylines[5];

    mylines[0].x1 = -40 + 50;
    mylines[0].y1 = 70 + 50;
    mylines[0].x2 = -20 + 50;
    mylines[0].y2 = 100 + 50;

    mylines[1].x1 = -40 + 50;
    mylines[1].y1 = 20 + 50;
    mylines[1].x2 = -10 + 50;
    mylines[1].y2 = 70 + 50;

    mylines[2].x1 = -10 + 50;
    mylines[2].y1 = 50 + 50;
    mylines[2].x2 = 30 + 50;
    mylines[2].y2 = 80 + 50;

    mylines[3].x1 = -20 + 50;
    mylines[3].y1 = 30 + 50;
    mylines[3].x2 = 10 + 50;
    mylines[3].y2 = 20 + 50;

    mylines[4].x1 = 10 + 50;
    mylines[4].y1 = -20 + 50;
    mylines[4].x2 = 30 + 50;
    mylines[4].y2 = 30 + 50;

    for (int i = 0; i < 5; i++) {
        line(mylines[i].x1,
mylines[i].y1, mylines[i].x2,
mylines[i].y2);
        delay(1000);
    }

    for (int i = 0; i < 5; i++) {
        clip(mylines[i]);
        delay(1000);
    }

    delay(40000);
    closegraph();
    return 0;
}
```

Screenshot:



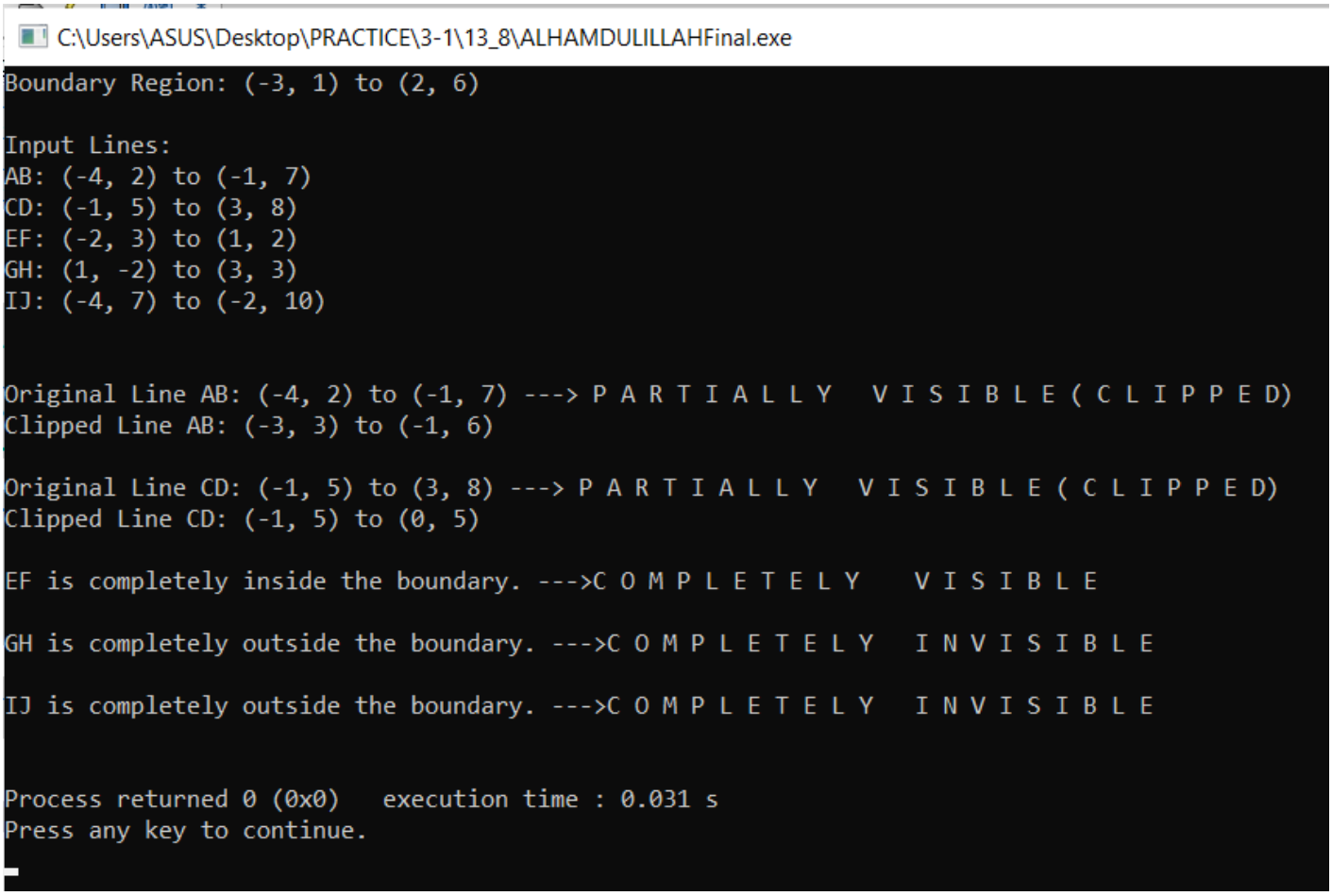
Source Code:

```
#include <iostream>

void printBoundary(int xMin, int
xMax, int yMin, int yMax) {
    std::cout << "Boundary Region:
(" << xMin << ", " << yMin << ") to
(" << xMax << ", " << yMax << ")" <<
std::endl;
}

void printInputLines(int lines[][4],
const std::string lineNames[], int
numLines) {
    std::cout << "Input Lines:" <<
std::endl;
    for (int i = 0; i < numLines;
i++) {
        std::cout << lineNames[i] <<
": (" << lines[i][0] << ", " <<
lines[i][1] << ") to (" <<
lines[i][2] << ", " << lines[i][3]
<< ")" << std::endl;
    }
    std::cout << std::endl;
}

void liangBarskyClip(int x0, int y0,
int x1, int y1, int xMin, int xMax,
int yMin, int yMax, const
std::string& lineName) {
    int p1 = -(x1 - x0);
    int p2 = -p1;
    int p3 = -(y1 - y0);
    int p4 = -p3;

    int q1 = x0 - xMin;
    int q2 = xMax - x0;
    int q3 = y0 - yMin;
    int q4 = yMax - y0;

    if (p1 == 0 && q1 < 0) {
        std::cout << lineName << "
is completely outside the boundary."
<< " --->C O M P L E T E L Y   I N V
I S I B L E"<< std::endl;
        return;
    }
    if (p3 == 0 && q3 < 0) {
        std::cout << lineName << "
is completely outside the boundary."
<< " --->C O M P L E T E L Y   I N V
I S I B L E"<< std::endl;
        return;
    }

    double u1 = 0, u2 = 1;

    if (p1 != 0) {
        double r1 = (double)q1 / p1;
        double r2 = (double)q2 / p2;
        if (p1 < 0) {
            u1 = std::max(u1, r1);
            u2 = std::min(u2, r2);
        } else {
            u1 = std::max(u1, r2);
            u2 = std::min(u2, r1);
        }
    }

    if (p3 != 0) {
        double r3 = (double)q3 / p3;
        double r4 = (double)q4 / p4;
        if (p3 < 0) {
            u1 = std::max(u1, r3);
            u2 = std::min(u2, r4);
        } else {
            u1 = std::max(u1, r4);
            u2 = std::min(u2, r3);
        }
    }

    if (u1 > u2) {
        std::cout << lineName << "
is completely outside the boundary."
<< " --->C O M P L E T E L Y   I N V
I S I B L E"<< std::endl;
        return;
    }

    if (u1 == 0 && u2 == 1) {
        std::cout << lineName << "
is completely inside the boundary."
<< " --->C O M P L E T E L Y    V I
S I B L E"<< std::endl;
```

```
        return;
    }

    int clippedX0 = x0 + u1 * (x1 - x0);
    int clippedY0 = y0 + u1 * (y1 - y0);
    int clippedX1 = x0 + u2 * (x1 - x0);
    int clippedY1 = y0 + u2 * (y1 - y0);

    std::cout << "Original Line " << lineName << ": (" << x0 << ", " << y0 << ") to (" << x1 << ", " << y1 << ")" << " ---> P A R T I A L L Y V I S I B L E ( C L I P P E D)"<< std::endl;
    std::cout << "Clipped Line " << lineName << ": (" << clippedX0 << ", " << clippedY0 << ") to (" << clippedX1 << ", " << clippedY1 << ")" << std::endl;
}
```


```
        return;
    }

    int clippedX0 = x0 + u1 * (x1 - x0);
    int clippedY0 = y0 + u1 * (y1 - y0);
    int clippedX1 = x0 + u2 * (x1 - x0);
    int clippedY1 = y0 + u2 * (y1 - y0);

    std::cout << "Original Line " << lineName << ": (" << x0 << ", " << y0 << ") to (" << x1 << ", " << y1 << ")" << " ---> P A R T I A L L Y V I S I B L E ( C L I P P E D)"<< std::endl;
    std::cout << "Clipped Line " << lineName << ": (" << clippedX0 << ", " << clippedY0 << ") to (" << clippedX1 << ", " << clippedY1 << ")" << std::endl;
}
```

Screenshot:


C:\Users\ASUS\Desktop\PRACTICE\3-1\13_8\ALHAMDULILLAHFinal.exe
Boundary Region: (-3, 1) to (2, 6)
Input Lines:
AB: (-4, 2) to (-1, 7)
CD: (-1, 5) to (3, 8)
EF: (-2, 3) to (1, 2)
GH: (1, -2) to (3, 3)
IJ: (-4, 7) to (-2, 10)
Original Line AB: (-4, 2) to (-1, 7) ---> P A R T I A L L Y V I S I B L E (C L I P P E D)
Clipped Line AB: (-3, 3) to (-1, 6)
Original Line CD: (-1, 5) to (3, 8) ---> P A R T I A L L Y V I S I B L E (C L I P P E D)
Clipped Line CD: (-1, 5) to (0, 5)
EF is completely inside the boundary. --->C O M P L E T E L Y V I S I B L E
GH is completely outside the boundary. --->C O M P L E T E L Y I N V I S I B L E
IJ is completely outside the boundary. --->C O M P L E T E L Y I N V I S I B L E
Process returned 0 (0x0) execution time : 0.031 s
Press any key to continue.

----------THE END----------