WORKSHEET 8

Student Name: Sumit Kumar **UID: 22BCS10048**

Section/Group: KRG_IOT-1-A **Semester: 6th Semester**

Branch: BE-CSE **Date of Performance: 20/03/2025**

Subject Name: Computer Graphics with Lab **Subject Code: 22CSH-352**

1. Aim: a). Apply the Cohen-Sutherland Line Clipping algorithm to clip a line intersecting at one point with a given window.
b). Apply the Cohen-Sutherland Line Clipping algorithm to clip a line intersecting at two or more points with a given window

2. Objective: To clip a line intersecting at a single point and two or more points with a window using the Cohen-Sutherland Line Clipping algorithm.

3. Implementation/Code:

a). For One Point

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

void main() {     int gd =
DETECT, gm;
   float i, xmax, ymax, xmin, ymin, x1, y1, x2, y2, m;
float start[4], end[4], code[4];

   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

   printf("\n\tEnter the bottom-left coordinate of viewport: ");
scanf("%f %f", &xmin, &ymin);

   printf("\n\tEnter the top-right coordinate of viewport: ");
   scanf("%f %f", &xmax, &ymax);

   printf("\nEnter the coordinates for starting point of line: ");
scanf("%f %f", &x1, &y1);
```

```
    printf("\nEnter the coordinates for ending point of line: ");
    scanf("%f %f", &x2, &y2);

    // Initialize region codes
for (i = 0; i < 4; i++) {
start[i] = 0;        end[i] = 0;
    }

    // Check for vertical line case to avoid division by zero
if (x2 - x1 == 0) {
        m = 0; // Avoid division by zero
    } else {
        m = (y2 - y1) / (x2 - x1);
    }

    // Calculate region codes
if (x1 < xmin) start[0] = 1;
if (x1 > xmax) start[1] = 1;
if (y1 > ymax) start[2] = 1;
if (y1 < ymin) start[3] = 1;

    if (x2 < xmin) end[0] = 1;
if (x2 > xmax) end[1] = 1;     if
(y2 > ymax) end[2] = 1;     if
(y2 < ymin) end[3] = 1;

            if ((end[2] == 1) && (end[3] == 0)) {
x2 = x2 + (ymax - y2) / m;                y2 =
ymax;
            }
            if ((start[1] == 0) && (start[0] == 1)) {
y1 = y1 + m * (xmin - x1);                x1 =
xmin;
            }
            if ((end[1] == 0) && (end[0] == 1)) {
y2 = y2 + m * (xmin - x2);                x2 =
xmin;
            }
```

```
            // Display clipped line
cleardevice();              printf("\n\t\tAfter
clipping:");              rectangle(xmin, ymin,
xmax, ymax);
            line(x1, y1, x2, y2);
            getch();
        }
    } else {
        // Case 3: Line is completely outside
        cleardevice();
        printf("\nLine is completely outside the viewport.");
rectangle(xmin, ymin, xmax, ymax);
    }

    getch();
closegraph();
}
```

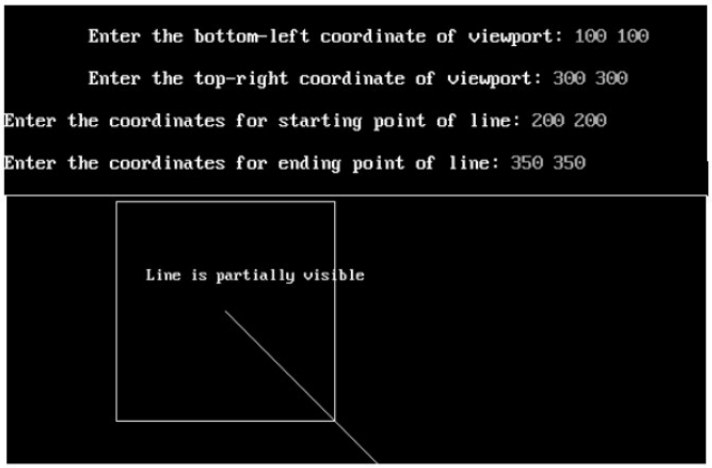
b). For Two Points #include
<stdio.h>
#include <conio.h>
#include <graphics.h>

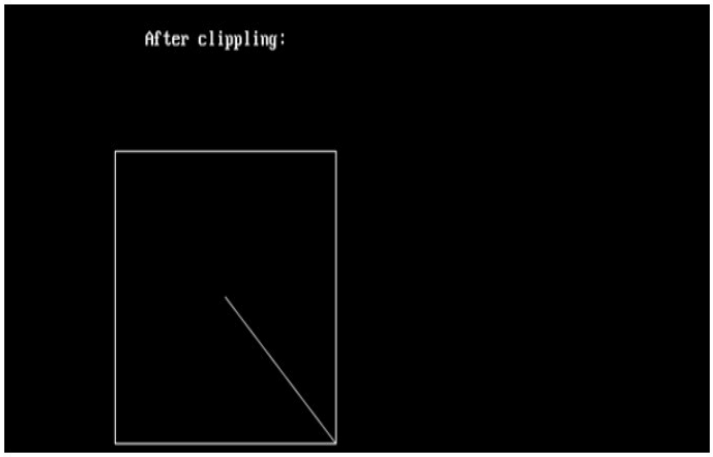
```
void main() {     int gd =
DETECT, gm;
    float i, xmax, ymax, xmin, ymin, x1, y1, x2, y2, m;
    float start[4], end[4], code[4];
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
    printf("\n\tEnter the bottom-left coordinate of viewport: ");
    scanf("%lf %lf", &xmin, &ymin);
    printf("\n\tEnter the top-right coordinate of viewport: ");
    scanf("%lf %lf", &xmax, &ymax);
    printf("\nEnter the coordinates for starting point of line: ");
    scanf("%lf %lf", &x1, &y1);
    printf("\nEnter the coordinates for ending point of line: ");
    scanf("%lf %lf", &x2, &y2);
    // Handle vertical line case to prevent division by zero
if (x2 - x1 == 0) {         m = 0;     } else {
```

```
        m = (y2 - y1) / (x2 - x1);
    }
    if (x1 < xmin) start[0] = 1;
if (x1 > xmax) start[1] = 1;     if
(y1 > ymax) start[2] = 1;
    if (y1 < ymin) start[3] = 1;
    if (x2 < xmin) end[0] = 1;
if (x2 > xmax) end[1] = 1;     if
(y2 > ymax) end[2] = 1;
    if (y2 < ymin) end[3] = 1;
    for (i = 0; i < 4; i++)
        code[i] = start[i] & end[i]; // Fixed bitwise operation

    if ((code[0] == 0) && (code[1] == 0) && (code[2] == 0) && (code[3] == 0)) {
if ((start[0] == 0) && (start[1] == 0) && (start[2] == 0) && (start[3] == 0) &&
(end[0] == 0) && (end[1] == 0) && (end[2] == 0) && (end[3] == 0)) {
cleardevice();
            printf("\n\t\tThe line is totally visible\n\t\tand not a clipping candidate");
rectangle(xmin, ymin, xmax, ymax);
            line(x1, y1, x2, y2);
            getch();         } else {
cleardevice();             printf("\n\t\tLine is
partially visible");             rectangle(xmin,
ymin, xmax, ymax);
            line(x1, y1, x2, y2);
            getch();

            // Display clipped line
cleardevice();             printf("\n\t\tAfter
clipping:");             rectangle(xmin, ymin,
xmax, ymax);
    } else {         cleardevice();
printf("\nLine is invisible");
        rectangle(xmin, ymin, xmax, ymax);
    }

    getch();
closegraph();
}
```

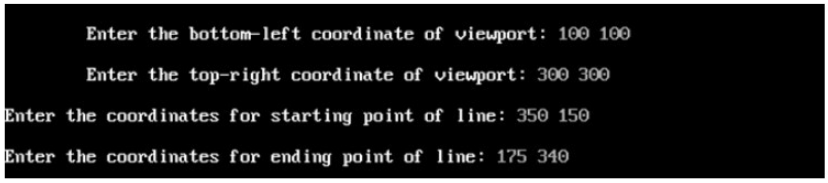
4. Output

a). For One Points

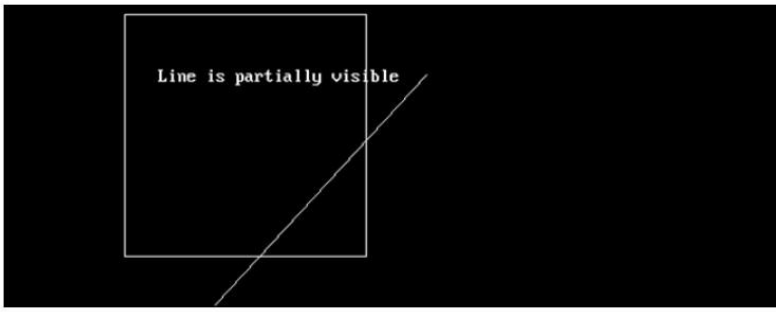


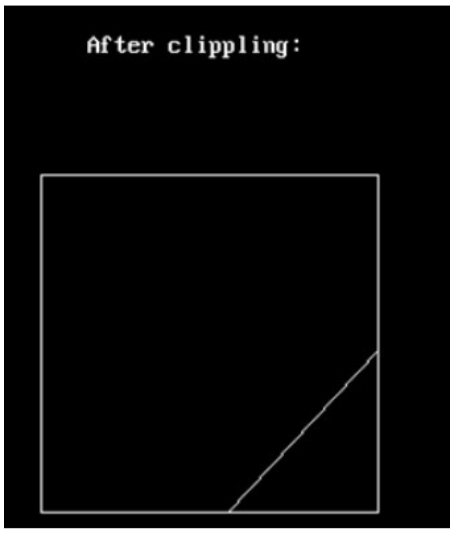


a).



b).





5. Learning Outcome

- Understanding Line Clipping – Learned Cohen-Sutherland algorithm, region codes, and clipping logic.
- Debugging C Graphics – Fixed division by zero, logical errors, and improved code efficiency.

Graphics Implementation – Used Turbo C++ functions to create and modify viewport-based line rendering.