Name:	MAHEK SHAH
Roll No:	
Class/Sem:	SE/III
Experiment No.:	8
Title:	Sutherland Hodgeman Polygon Clipping Algorithm
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Experiment No.8

Aim :Write a program to implement Sutherland Hodgeman Polygon Clipping Algorithm in C.

Objective :To implement Sutherland Hodgeman Polygon Clipping algorithm for clipping a subject polygon with respect to a clip polygon.

Sutherland-Hodgeman Polygon Clipping:
It is performed by processing the boundary of polygon against each window corner or edge. First of all entire polygon is clipped against one edge, then resulting polygon is considered, then the polygon is considered against the second edge, so on for all four edges.

Four possible situations while processing

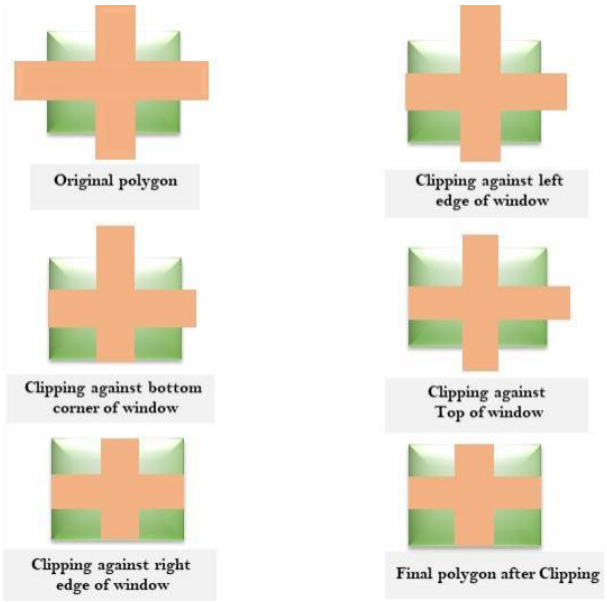
If the first vertex is an outside the window, the second vertex is inside the window. Then second vertex is added to the output list. The point of intersection of window boundary and polygon side (edge) is also added to the output line.

If both vertexes are inside window boundary. Then only second vertex is added to the output list.

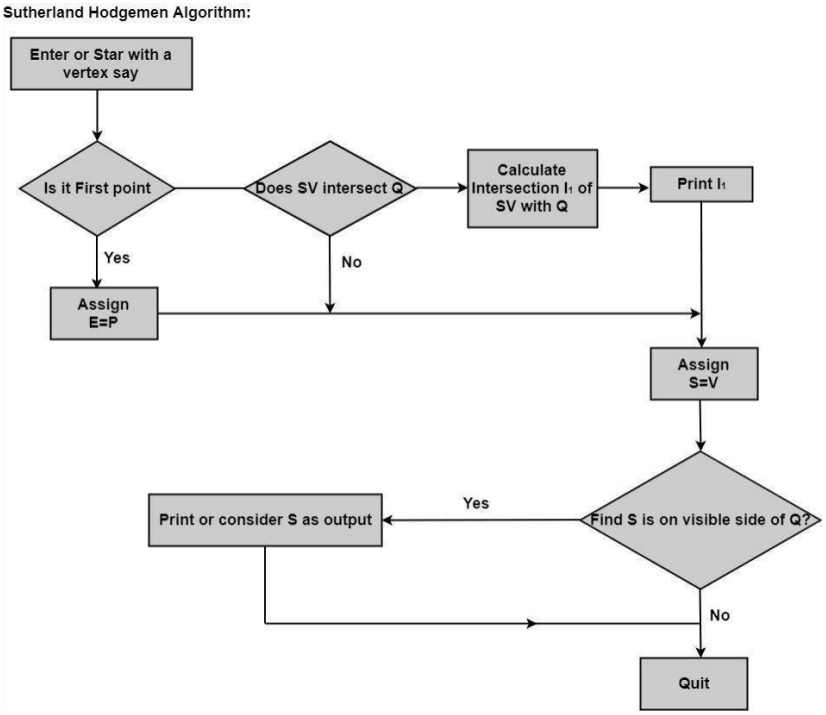
If the first vertex is inside the window and second is an outside window. The edge which intersects with window is added to output list.

If both vertices are the outside window, then nothing is added to output list.

Following figures shows original polygon and clipping of polygon against four windows.



Algorithm:



Code :

```
#include <stdio.h>
#include <graphics.h>
int subjectPolygon[][2] = {{50, 150}, {150, 150}, {150, 50}, {50, 50}};
int clipPolygon[][2] = {{100, 100}, {200, 100}, {200, 0}, {100, 0}};
int inside(int p[2], int clipEdge[2][2], int edge) {
    return (clipEdge[1][0] - clipEdge[0][0]) * (p[1] - clipEdge[0][1]) >
           (clipEdge[1][1] - clipEdge[0][1]) * (p[0] - clipEdge[0][0]);
}
void computeIntersection(int p1[2], int p2[2], int clipEdge[2][2], int edge,
int intersection[2]) {
    int dx = p2[0] - p1[0];
    int dy = p2[1] - p1[1];
    int numerator = (clipEdge[0][1] - p1[1]) * dx - (clipEdge[0][0] - p1[0]) *
dy;
    int denominator = (clipEdge[1][0] - clipEdge[0][0]) * dy - (clipEdge[1][1]
- clipEdge[0][1]) * dx;
    intersection[0] = p1[0] + numerator / denominator;
    intersection[1] = p1[1] + numerator / denominator;
}
```

```
void sutherlandHodgman() {
    int outputPolygon[4][2];
    int inputPolygonSize = 4, clipEdgeSize = 4;
    for (int edge = 0; edge < clipEdgeSize; edge++) {
        int inputSize = inputPolygonSize;
        int inputPolygon[inputPolygonSize][2];
        for (int i = 0; i < inputPolygonSize; i++) {
            inputPolygon[i][0] = subjectPolygon[i][0];
            inputPolygon[i][1] = subjectPolygon[i][1];
        }
        int outputSize = 0;
        int clipEdge[2][2] = {{clipPolygon[edge][0], clipPolygon[edge][1]},
                              {clipPolygon[(edge + 1) % clipEdgeSize][0],
clipPolygon[(edge + 1) % clipEdgeSize][1]}};
        for (int i = 0; i < inputSize; i++) {
            int p1[2] = {inputPolygon[i][0], inputPolygon[i][1]};
            int p2[2] = {inputPolygon[(i + 1) % inputSize][0], inputPolygon[(i
+ 1) % inputSize][1]};
            if (inside(p2, clipEdge, edge)) {
                if (!inside(p1, clipEdge, edge)) {
                    computeIntersection(p1, p2, clipEdge, edge,
outputPolygon[outputSize]);
                    outputSize++;
                }
                outputPolygon[outputSize][0] = p2[0];
                outputPolygon[outputSize][1] = p2[1];
                outputSize++;
            } else if (inside(p1, clipEdge, edge)) {
                computeIntersection(p1, p2, clipEdge, edge,
outputPolygon[outputSize]);
                outputSize++;
            }
        }
        inputPolygonSize = outputSize;
        for (int i = 0; i < outputSize; i++) {
            subjectPolygon[i][0] = outputPolygon[i][0];
            subjectPolygon[i][1] = outputPolygon[i][1];
        }
    }
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    setcolor(RED);
    drawpoly(inputPolygonSize, (int *)subjectPolygon);
    setcolor(GREEN);
    drawpoly(clipEdgeSize, (int *)clipPolygon);
```

```
    delay(5000);
    closegraph();
}
int main() {
    sutherlandHodgman();
    return 0;
}
```

Output :

Conclusion :

The Sutherland-Hodgman polygon clipping algorithm is a simple and effective method for clipping polygons against an arbitrary convex clip window. The algorithm works by iteratively clipping the polygon against each edge of the clip window.

The implementation involves checking whether each vertex of the polygon is inside or outside the clip window. If a vertex is inside, it is included in the output. If it is outside, the algorithm computes the intersection point of the polygon edge with the clip window edge and includes the intersection point in the output.

This algorithm is particularly useful in computer graphics for removing parts of a polygon that lie outside a specified viewing area, which is common in rendering and display processes. While Sutherland-Hodgman is effective for convex clip windows, it may require modification for handling concave clip windows or more complex cases. Additionally, more advanced algorithms, such as the Cyrus-Beck algorithm or the Weiler-Atherton algorithm, are employed for handling more general cases of polygon clipping.