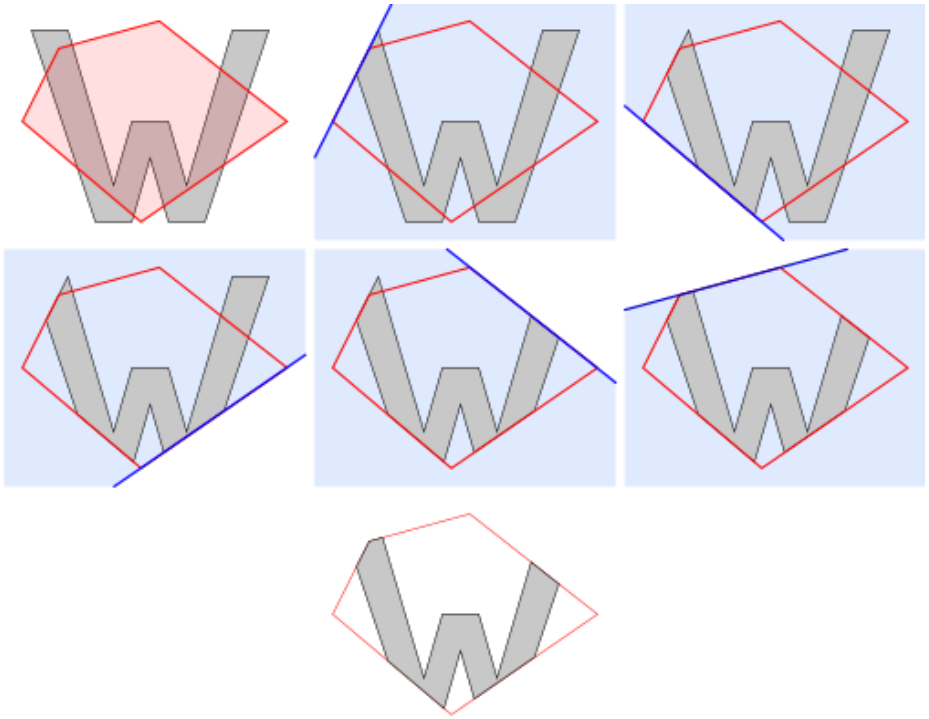
# Program 10

**Aim:** Write a program to clip a Polygon using Sutherland Hodgeman Algorithm.

## Theory:

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. The algorithm begins with an input list of all vertices in the subject polygon. Next, one side of the clip polygon is extended infinitely in both directions, and the path of the subject polygon is traversed. Vertices from the input list are inserted into an output list if they lie on the visible side of the extended clip polygon line, and new vertices are added to the output list where the subject polygon path crosses the extended clip polygon line.

This process is repeated iteratively for each clip polygon side, using the output list from one stage as the input list for the next. Once all sides of the clip polygon have been processed, the final generated list of vertices defines a new single polygon that is entirely visible. Note that if the subject polygon was concave at vertices outside the clipping polygon, the new polygon may have coincident (i.e., overlapping) edges – this is acceptable for rendering, but not for other applications such as computing shadows.



## Pseudocode:

```
List outputList = subjectPolygon;

for (Edge clipEdge in clipPolygon) do
    List inputList = outputList;
    outputList.clear();

    for (int i = 0; i < inputList.count; i += 1) do
        Point current_point = inputList[i];
        Point prev_point = inputList[(i - 1) % inputList.count];

        Point Intersecting_point = ComputeIntersection(prev_point, current_point, clipEdge)

        if (current_point inside clipEdge) then
            if (prev_point not inside clipEdge) then
                outputList.add(Intersecting_point);
            end if
            outputList.add(current_point);

        else if (prev_point inside clipEdge) then
            outputList.add(Intersecting_point);
        end if

    done
done
```

## Code:

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

int round(int a){
    return ((int)(a+0.5));
}

float xmin, ymin, xmax, ymax, arr[20], m; int k;
void left_clip(float x1, float y1, float x2, float y2){
    if (x2 - x1)
        m = (y2 - y1) / (x2 - x1);
    else
        m = 100000;
    if(x1 >= xmin && x2 >= xmin){
        arr[k] = x2;
        arr[k + 1] = y2;
```

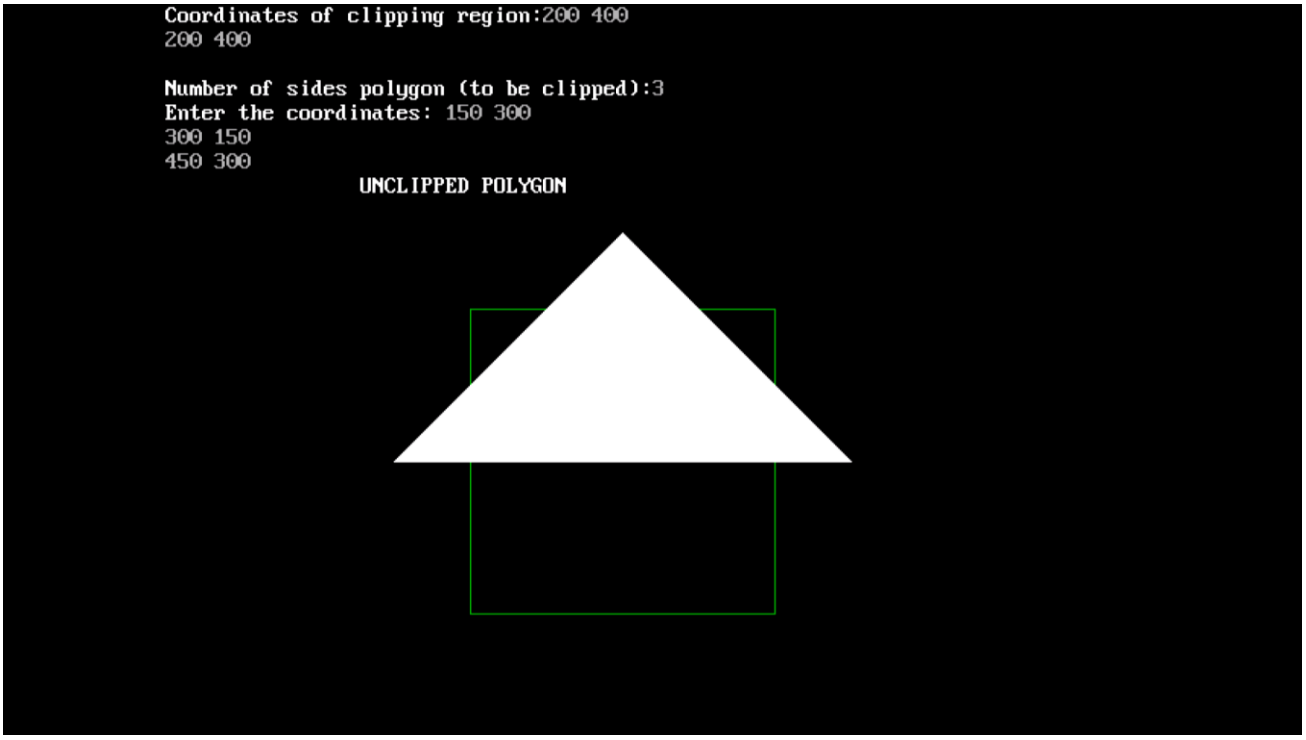
```
        k += 2;
    }
    if(x1 < xmin && x2 >= xmin){
        arr[k] = xmin;
        arr[k + 1] = y1 + m * (xmin - x1);
        arr[k + 2] = x2;
        arr[k + 3] = y2;
        k += 4;
    }
    if(x1 >= xmin && x2 < xmin){
        arr[k] = xmin;
        arr[k + 1] = y1 + m * (xmin - x1);
        k += 2;
    }
}

void top_clip(float x1, float y1, float x2, float y2){
    if (y2 - y1)
        m = (x2 - x1) / (y2 - y1);
    else
        m = 100000;
    if (y1 <= ymax && y2 <= ymax){
        arr[k] = x2;
        arr[k + 1] = y2;
        k += 2;
    }
    if (y1 > ymax && y2 <= ymax){
        arr[k] = x1 + m * (ymax - y1);
        arr[k + 1] = ymax;
        arr[k + 2] = x2;
        arr[k + 3] = y2;
        k += 4;
    }
    if (y1 <= ymax && y2 > ymax){
        arr[k] = x1 + m * (ymax - y1);
        arr[k + 1] = ymax;
        k += 2;
    }
}

void right_clip(float x1, float y1, float x2, float y2){
    if(x2 - x1)
```

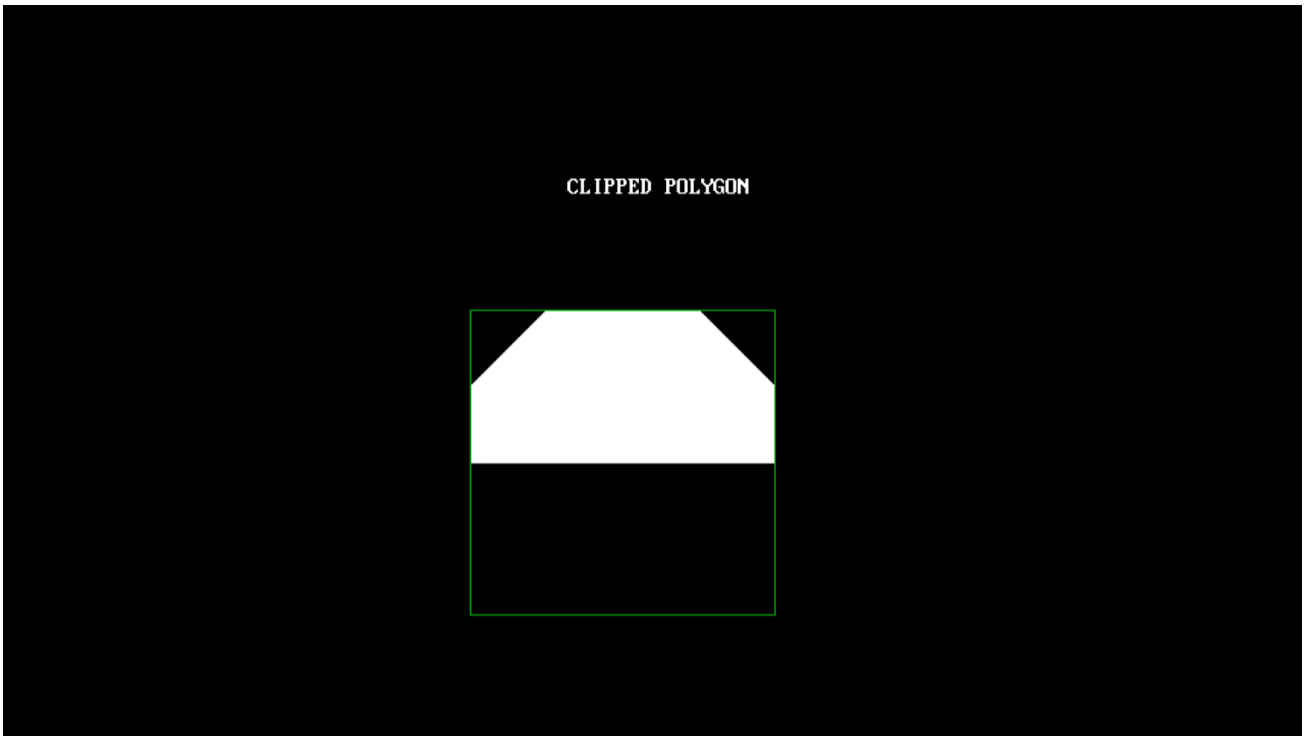
```
        m = (y2 - y1) / (x2 - x1);
    else
        m = 100000;
    if(x1 <= xmax && x2 <= xmax){
        arr[k] = x2;
        arr[k + 1] = y2;
        k += 2;
    }
    if(x1 > xmax && x2 <= xmax){
        arr[k] = xmax;
        arr[k + 1] = y1 + m * (xmax - x1);
        arr[k + 2] = x2;
        arr[k + 3] = y2;
        k += 4;
    }
    if(x1 <= xmax && x2 > xmax){
        arr[k] = xmax;
        arr[k + 1] = y1 + m * (xmax - x1);
        k += 2;
    }
}

void bottom_clip(float x1, float y1, float x2, float y2){
    if(y2 - y1)
        m = (x2 - x1) / (y2 - y1);
    else
        m = 100000;
    if(y1 >= ymin && y2 >= ymin){
        arr[k] = x2;
        arr[k + 1] = y2;
        k += 2;
    }
    if(y1 < ymin && y2 >= ymin){
        arr[k] = x1 + m * (ymin - y1);
        arr[k + 1] = ymin;
        arr[k + 2] = x2;
        arr[k + 3] = y2;
        k += 4;
    }
    if(y1 >= ymin && y2 < ymin){
        arr[k] = x1 + m * (ymin - y1);
        arr[k + 1] = ymin;
```

```
        k += 2;
    }
}

void main(){

    int g_driver = DETECT, g_mode;
    initgraph(&g_driver, &g_mode, "C:\\TURBOC3\\BGI");

    float xi, yi, xf, yf, polyy[20];
    int n, poly[20];
    //clrscr();
    cout<<"Coordinates of clipping region:";
    cin>>xmin>>xmax;
    cin>>ymin>>ymax;
    cout<<"\nNumber of sides polygon (to be clipped):";
    cin>>n;
    cout<<"Enter the coordinates: ";
    for(int i = 0; i < 2 * n; i++)
        cin>>polyy[i];
    polyy[i] = polyy[0];
    polyy[i + 1] = polyy[1];
    for (i = 0; i < 2 * n + 2; i++)
        poly[i] = round(polyy[i]);
    setcolor(GREEN);
    rectangle(xmin, ymax, xmax, ymin);
    cout<<"\t\tUNCLIPPED POLYGON";
    setcolor(WHITE);
    fillpoly(n, poly);
    getch();
    cleardevice();
    k = 0;
    for(i=0; i<2*n; i+=2)
        left_clip(polyy[i], polyy[i+ 1], polyy[i + 2], polyy[i
+ 3]);
    n = k / 2;
    for(i=0; i<k; i++)
        polyy[i] = arr[i];
    polyy[i] = polyy[0];
    polyy[i+1] = polyy[1];
    k=0;
    for(i=0; i<2*n; i+=2)
```

```
        top_clip(polyy[i], polyy[i+1], polyy[i+2], polyy[i+3]);
    n=k/2;
    for(i=0; i<k; i++)
        polyy[i] = arr[i];
    polyy[i] = polyy[0];
    polyy[i+1] = polyy[1];
    k=0;
    for(i=0;i<2*n;i+=2)
        right_clip(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);
    n=k/2;
    for(i=0;i<k;i++)
        polyy[i]=arr[i];
    polyy[i]=polyy[0];
    polyy[i+1]=polyy[1];
    k = 0;
    for(i=0; i<2*n; i+=2)
        bottom_clip(polyy[i], polyy[i+1], polyy[i+2], polyy[i+3
]);
    for(i=0; i<k; i++)
        poly[i]=round(arr[i]);
    if(k)
        fillpoly(k/2,poly);
    setcolor(GREEN);
    rectangle(xmin, ymax, xmax, ymin);
    cout<<"CLIPPED POLYGON";
    getch();
    closegraph();
}
```

# Output:


Coordinates of clipping region:200 400
200 400
Number of sides polygon (to be clipped):3
Enter the coordinates: 150 300
300 150
450 300
UNCLIPPED POLYGON


CLIPPED POLYGON