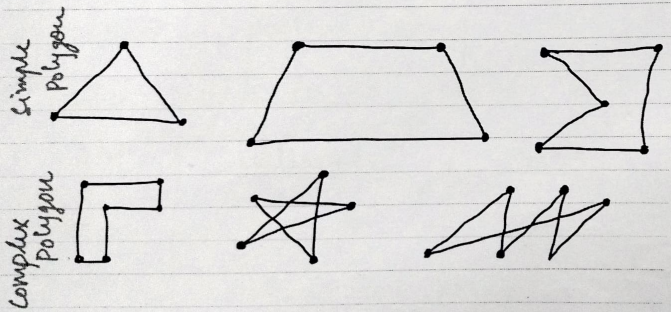
POLYGON

Properties of Polygon

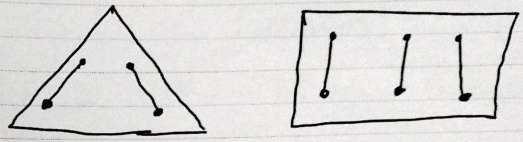
↳ It consist of line segment.

↳ It must be closed circuit.



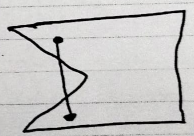
Types of Polygon

① Convex Polygon :- If the two interior points of the lines lies inside the polygon then it is convex.



② Concave Polygon :- If the two interior points of the lines lies outside the polygon then it is called concave polygon.

Ex :-



Remarks :- If the angle of polygon is less than 180° then it is called as convex polygon.

Else, It is called concave greater than 180°.

Inside Test (Polygon)

↳ It is used to identify the pixel/point is within the polygon or outside the polygon.

Algorithm:- Ray Casting

① Draw the horizontal line from the point to cover max vertices.

② Count the number of times the lines intersects with the edge.

③ If the number of count is number = odd

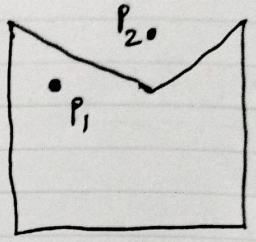
then it is inside

else,

number == even

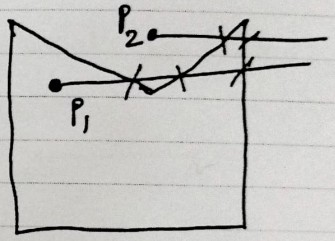
then it is outside.

Q.



Check the point is inside or outside of Polygon using Ray casting Algo

Sol^{n}:-



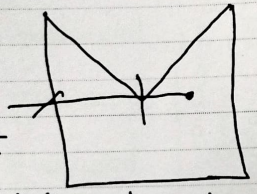
Point	Intersection	O/P
P_1	3	inside
P_2	2	outside

Special case:- If the line pass/cross at the vertex or cross-section point of two lines.

· We used the winding Number Algorithm

① Check the respective line segment of that point.

② Check the other end of the line segment.



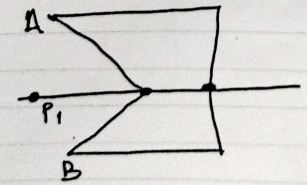
③ Check if both point is in same side 'or' different side of the line segment.

④ If the point is on same side then we consider as even.

If the point is not on same side then we consider as odd

Sol :- Not on the same side = odd

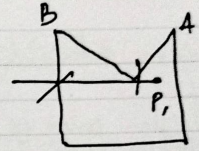
Intersection = 1



Then, = 1 + odd = even no of intersection

Conclusion :- Then point is outside.

If.



same side = even

Intersection = 1

Then = 1 + even = odd

Conclusion = odd no of Intersection.

Therefore point is inside.

Polygon Boundary Fill Algorithm

↳ Color Fill Algorithm.

↳ Boundary color is different

↳ within the polygon color is different.

Algorithm:-

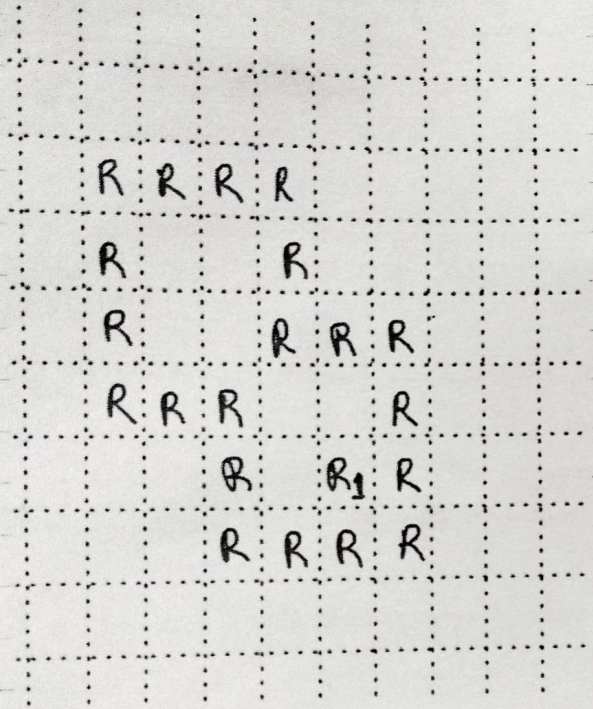
1. Let 'P' take the point inside the polygon.
2. Start with co-ordinate (x, y)
3. Inside color is 'F' ← Fill color.
4. Boundary color is 'b'

Algorithm steps:-

```
boundary (x, y, F, b)
{
if (getpixel (x, y) != b) ||
   (getpixel (x, y) != F)
{
putpixel (x, y, F)
boundary (x+1, y, F, b)
boundary (x, y+1, F, b)
boundary (x-1, y, F, b)
boundary (x, y-1, F, b)
boundary (x-1, y-1, F, b)
boundary (x-1, y+1, F, b)
boundary (x+1, y+1, F, b)
boundary (x+1, y-1, F, b)
}}
```

4-connected pixel is used to check the co-ordinates

8-connected Pixel Algorithm



(x-1, y-1)	(x, y+1)	(x+1, y+1)
(x-1, y)	(x, y)	(x+1, y)
(x-1, y-1)	(x, y-1)	(x+1, y-1)

8- connected Pixel

	(x, y+1)	
(x-1, y)	(x, y)	(x+1, y)
	(x, y-1)	

4 - connected Pixel

Flood fill Algorithm:-

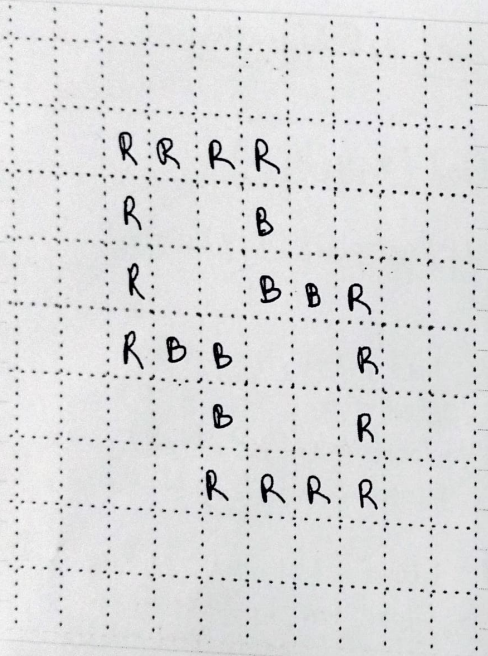
```
Flood (x, y, N, θ)
{
  if (getpixel(x, y) == θ)
  {
    Flood (x+1, y, N, θ)
    Flood (x, y+1, N, θ)
    Flood (x-1, y, N, θ)
    Flood (x, y-1, N, θ)

    Flood (x-1, y-1, N, θ)
    Flood (x-1, y+1, N, θ)
    Flood (x+1, y-1, N, θ)
    Flood (x-1, y+1, N, θ)
  }
}
```

4- connected pixel

8- connected Pixel

Here, N = New Color.

θ = Old Color.



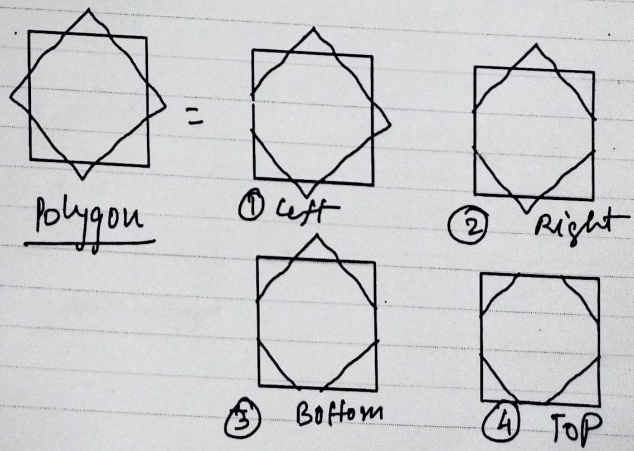
Sutherland Hodgeman Polygon clipping Algorithm

↳ It is used for clipping polygon.

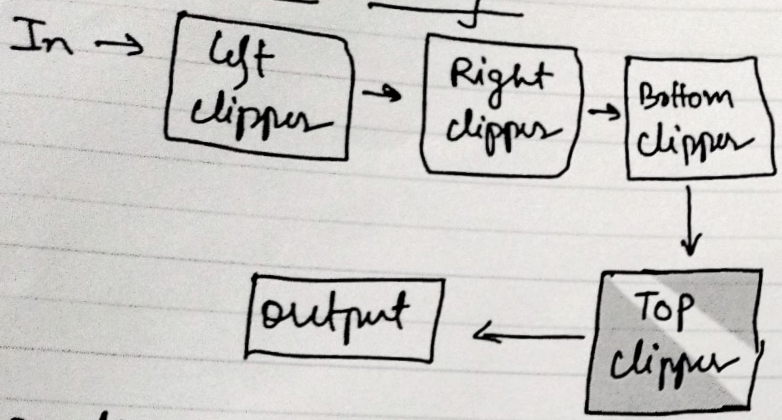
↳ In this we have given a polygon & window.

↳ Testing of Algorithm with respect to reference of Frame (RoF)

Ex1-



Algorithm Processing



→ clipping is done based on the boundary.

→ Edge is consider as boundary and respective inside and outside of point is consider for clipping.

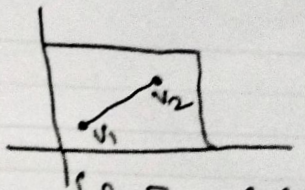
There are four cases for processing of vertices :-

Case 1 :- when both input vertices are inside the window boundary (WB)

Input = $v_1 v_2$

output = v_2

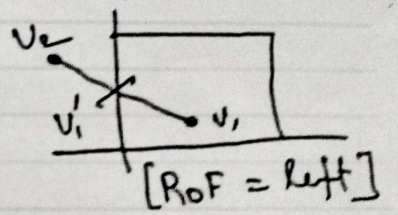
Consider 2nd vertices for the o/p.



Case 2 :- First vertices is inside and second is outside of (WB).

Input :- $v_1 v_2$

output :- v_1'

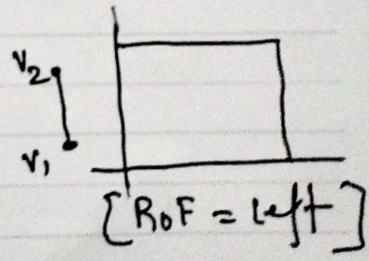


Consider the intersection point w.r.t first point.

Case 3 :- Both input vertices is outside the (WB).

Input :- $v_1 v_2$

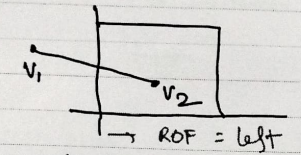
output :- Nil



Case 4:- If first vertices is outside and second vertices is inside of WB.

Input = V_1V_2

Output = V_1' & V_2

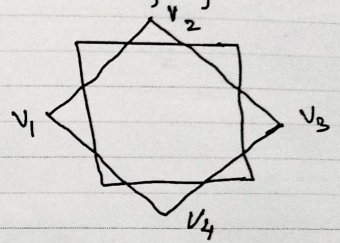


Take the intersection point of first point.

Based on the cases we have generate the truth table:-

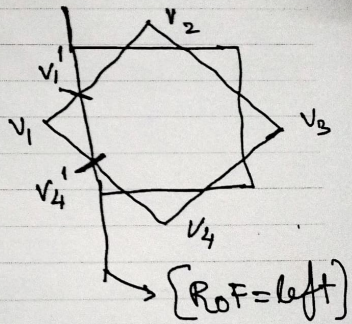
	Vertices V_1	Vertices V_2	O/P of vertices
①	In	In	V_2
②	out	out	Nil
③	In	out	V_1' (Intersection of First Point)
④	out	In	V_1' & V_2 (Intersection of first Point)

Q. Perform the clipping using Sutherland Hodgeman clipping Algorithm.



Sol^{n}:- Start from the [ROF = left]

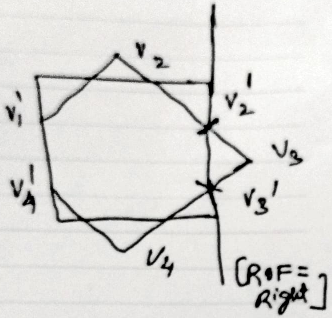
① ROF = Left



I/P	O/P
V_1V_2	= V_1' & V_2
V_2V_3	= V_3
V_3V_4	= V_4
V_4V_1	= V_4'

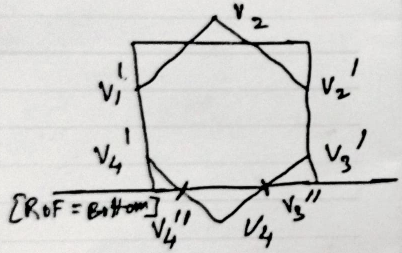
② ROF = Right

$V_1' V_2 = V_2$
$V_2 V_3 = V_2'$
$V_3 V_4 = V_3' \& V_4$
$V_4 V_4' = V_4'$
$V_4' V_1' = V_1'$



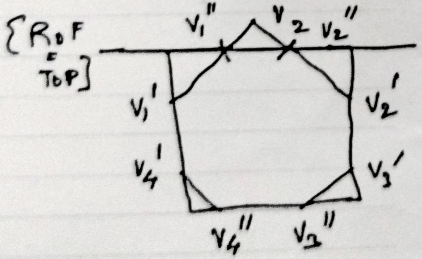
③ ROF = Bottom

$V_1' V_2 = V_2$
$V_2 V_2' = V_2'$
$V_2' V_3' = V_3'$
$V_3' V_4 = V_3''$
$V_4 V_4' = V_4'' \& V_4'$
$V_4' V_1' = V_1'$



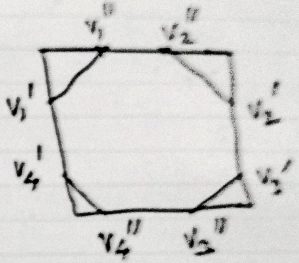
④ ROF = TOP

$V_1' V_2 = V_1''$
$V_2 V_2' = V_2'' \& V_2'$
$V_2' V_3' = V_3'$
$V_3' V_3'' = V_3''$
$V_3'' V_4'' = V_4''$
$V_4'' V_4' = V_4'$



$V_4' V_1' = V_1'$

O/P



I