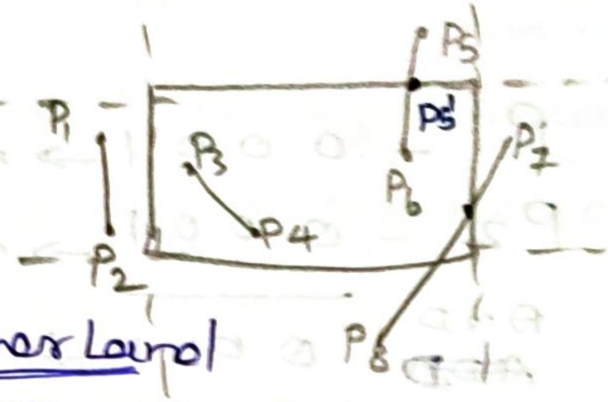
Line clipping: -



cohen-Sutherland Line clipping Algorithm.

$P_1 P_2 \rightarrow$ Reject

$P_3 P_4 \rightarrow$ Accept.

$P_5 P_6 \rightarrow$ clipping is Required.

$P_7 P_8 \rightarrow$ Cliping is Required.

Line clipping algm;

- cohen_Sutherland Algem.
- It is run based on Region code.
- Region Code is 4 bit code.
- Identify the neighbour.
- ABRL - To fill the bits.

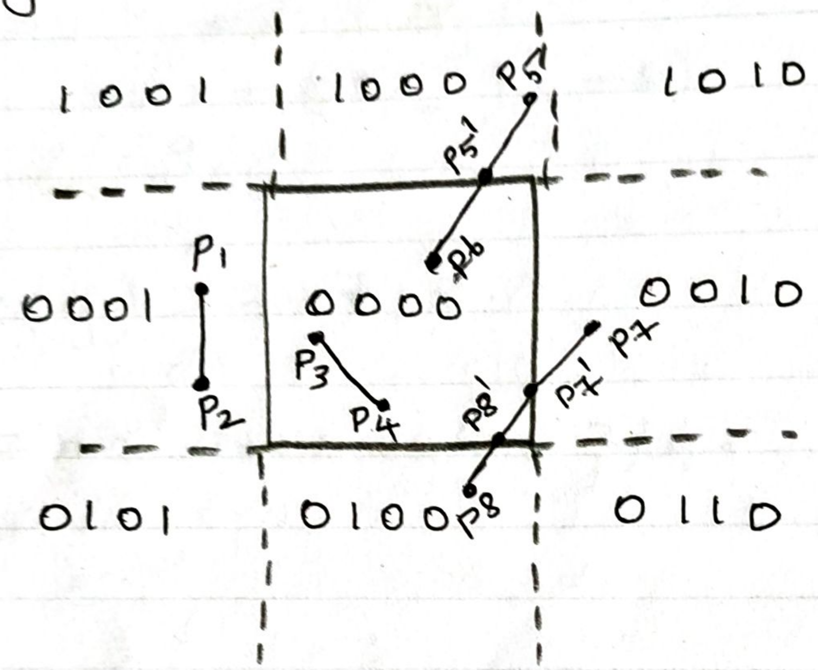
Above Below

1001	1000	1010
0001	0000	0010
0101	0100	0110

* Indentify the Region code.

Example :

✓ Cohen - Sutherland Line clipping Algorithm :



1. P1 & P2 :

P1 →	0001	Non Zero
P2 →	0001	non zero
AND	0001	Non zero. Reject.

if we get non zero, the Line is completly outside the window.

2. <u>P3 & P4 :</u>

P3 → 0 0 0 0 (zero)
P4 → 0 0 0 0 (zero)

AND → 0 0 0 0 (zero).

if both points are zero the points Lies inside the window.

* No clipping is required.
* Accept.

3. <u>P5 & P6 :</u>

P5 → 1 0 0 0 non zero
P6 → 0 0 0 0 zero.

AND 0 0 0 0 zero.

* Some portion of the points ~~Window~~ Lies inside the window and Some portion of the points Lies outside the Window.

* clipping is Required.
* Partial.

* Need to find intersection point.

P_5' & $P6$

P_5'	=	0000	Zero
$P6$	=	0000	Zero
AND		0000	Zero

P_5' & $P6$ Line Lies inside the window.

P_5 & P_5' is clipped.

P7 & P8:

P8	-	0100	Non Zero
P7	-	0010	Non Zero.
AND		0000	Zero.

Some portion inside the window and out side the window.

Find the Intersection point.

P7' and P7.

~~P7~~ P7' - 0 0 0 0
P7 - 0 0 1 0
And - 0 0 0 0

P8 & P8'

0 1 0 0
0 0 0 0
00 00

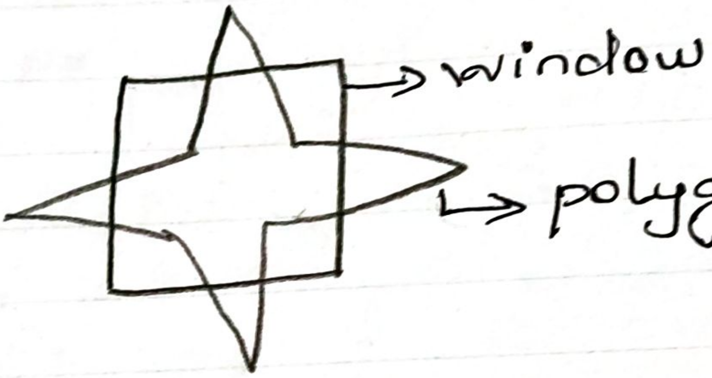
P8' - 0 0 0 0
P7' - 0 0 0 0
0 0 0 0.

clipped $\boxed{P8' \& P7'}$

$\boxed{P3 \& P4,\ P5' \& P6,\ P8',\ P7'}$

Sutherland - Hodgman Algorithm:

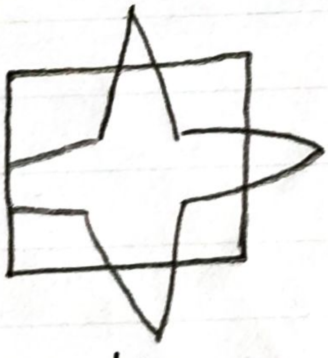
* The Sutherland - Hodgman algorithm is used for clipping polygons.
* In this algorithm, all the vertices of the polygon are clipped against each edge of the clipping window.



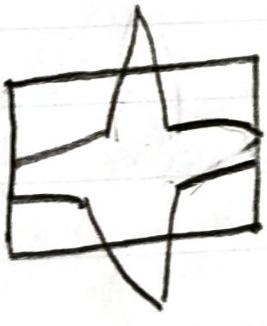
Initial polygon.

Steps for polygon clipping:

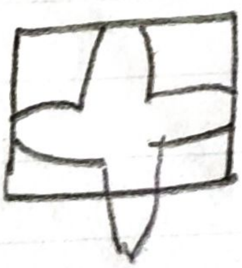
1. Left clip



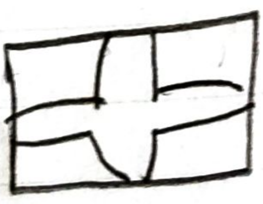
2. Right clip



3. Top clip

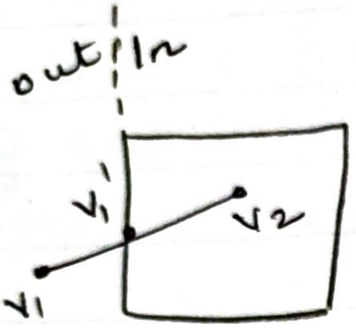


4. Bottom clip



Follow 4 cases:

case : 1

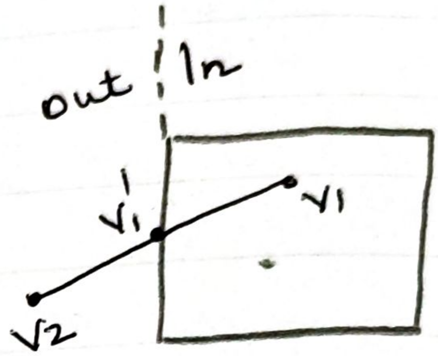


output :

Move out → in

(v_1' v_2) consider the points are

case : 2

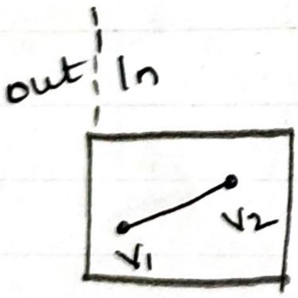


output :

Move In → out

(v_1')

case : 3

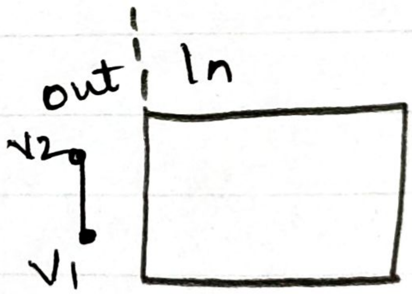


output :

Move In → In

v_2

case : 4



output :

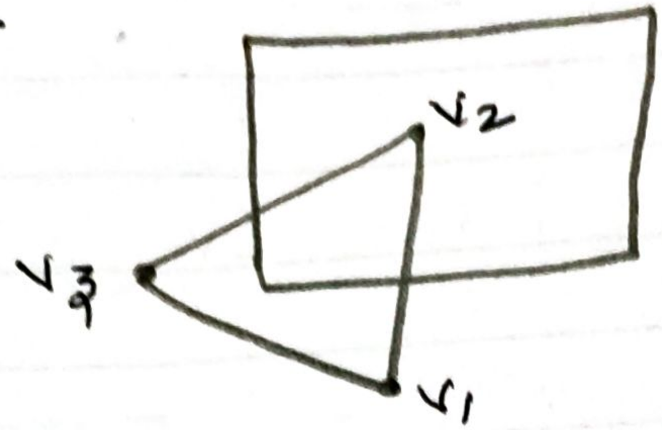
Move out → out

NIL.

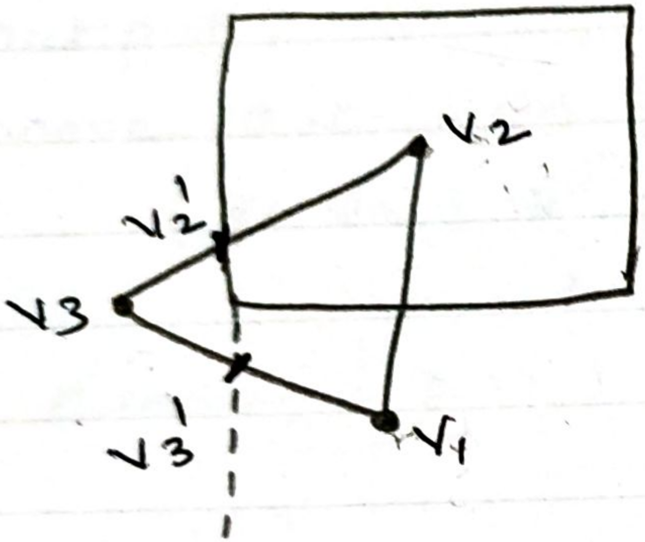
No need to consider any points.

~~Left~~

Example:



case 1: Left clip.

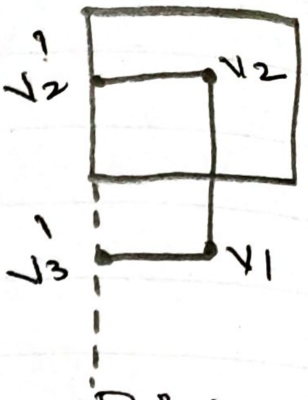


$V_1 V_2$ - In → In ⇒ V_2

$V_2 V_3$ - In → out ⇒ V_2'

$V_3 V_1$ - out → In ⇒ $V_3' V_1$

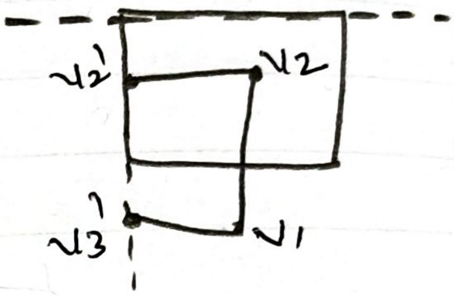
Case 2. Right clip:



$V_1 V_2$ — V_2 (Inside)
$V_2 V_2'$ — V_2' (Inside)
$V_2' V_3'$ — V_3' (Inside)
$V_3' V_1$ — V_1 (Inside)

No change after Right clip.

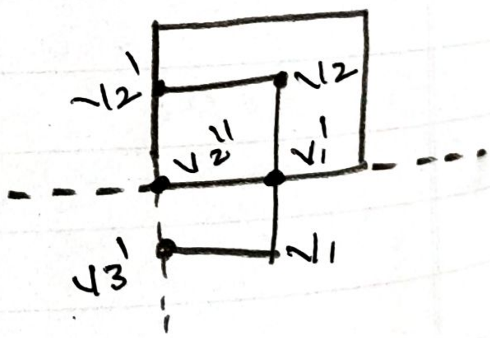
Case 3. Top clip:



$V_1 V_2$ — V_2 (Inside)
$V_2 V_2'$ — V_2' (Inside)
$V_2' V_3'$ — V_3' (Inside)
$V_3' V_1$ — V_1 (Inside)

No change after Top Clip.

Case 4. Bottom clip.



$V_1 V_2$ — $V_1' V_2$ (out – in)
$V_2 V_2'$ — V_2' (Inside)
$V_2' V_3$ — V_2'' (In to out)
$V_3' V_1$ — (out to out) NIL

Final output