Assignment-1

Name- Sudhanshu Raj

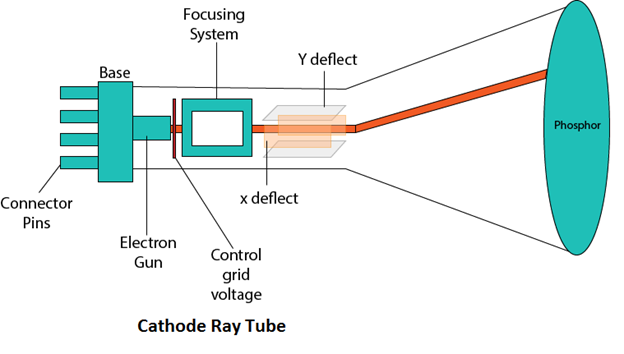
Sap id- 500097358

Enrollment no- R2142211331

Subject- Computer Graphic

1. **Explain the working of CRT? Also what happens in case of colour CRT.**

**Ans-** A cathode ray tube (CRT) is a vacuum tube that uses an electron beam to produce images on a fluorescent screen. It works by using an electron gun, which emits a beam of electrons from a heated cathode, through a series of focusing and deflecting electrodes, to create a moving image on the screen.



The electron gun generates a stream of electrons, which is accelerated by a high voltage anode towards the screen. The stream of electrons is focused into a narrow beam by a set of focusing electrodes, and then directed towards the screen by a set of deflecting electrodes. By varying the voltages on the deflecting electrodes, the beam can be moved horizontally and vertically to create an image.

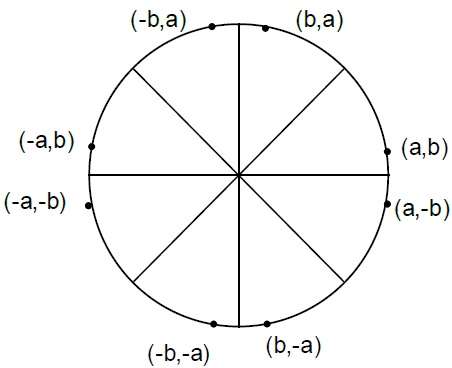
In a color CRT, there are three electron guns, one for each primary color: red, green, and blue. These electron guns are arranged in a triangular pattern, with each gun pointing towards a small dot on the screen. When the electron guns are activated, they emit three separate beams of electrons, each one aimed at a different dot on the screen.

The dots on the screen are coated with a phosphor that glows when it is hit by the electrons. By varying the intensity and position of each of the three electron beams, a wide range of colors can be produced on the screen.

To create a full-color image, the electron beams are rapidly switched on and off, in a process known as "scanning". The scanning process is synchronized with the signals from a video signal source, such as a television or computer. As the beams move across the screen, they create a series of lines, which, when viewed from a distance, form a complete image. The CRT technology was once widely used in televisions, computer monitors, and other display devices but has been largely replaced by newer display technologies such as LCD, OLED, and Plasma.

1. **Explain Circle Generation Algorithm.**

**Ans-** Circle Generation Algorithm refers to the process of generating the points that define a circle using a computer program. There are two algorithms Bresenham’s and midpoint circle algorithm that can be used to generate circles, but one of the most commonly used is the midpoint circle algorithm.



The equation of circle is X2+Y2=r2,

where r is radius.

Bresenham’s Algorithm

We cannot display a continuous arc on the raster display. Instead, we have to choose the nearest pixel position to complete the arc.

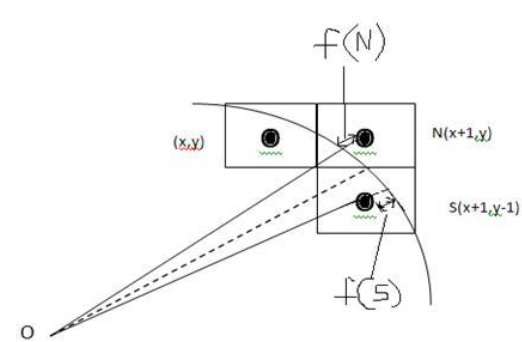
From the following illustration, you can see that we have put the pixel at X,Y location and now need to decide where to put the next pixel − at N X+1,Y or at S X+1,Y−1

This can be decided by the decision parameter **d**.

* If d <= 0, then N X+1,Y is to be chosen as next pixel.
* If d > 0, then S X+1,Y−1is to be chosen as the next pixel.

**Algorithm**

**Step 1** − Get the coordinates of the center of the circle and



radius, and store them in x, y, and R respectively.

Set P=0 and Q=R.

**Step 2** − Set decision parameter D = 3 – 2R.

**Step 3** − Repeat through step-8 while P ≤ Q.

**Step 4** − Call Draw Circle X,Y,P,Q

**Step 5** − Increment the value of P.

**Step 6** − If D < 0 then D = D + 4P + 6.

**Step 7** − Else Set R = R - 1, D = D + 4P−Q + 10.

**Step 8** − Call Draw Circle X,Y,P,Q.

Draw Circle Method(X, Y, P, Q).

Call Putpixel (X + P, Y + Q).

Call Putpixel (X - P, Y + Q).

Call Putpixel (X + P, Y - Q).

Call Putpixel (X - P, Y - Q).

Call Putpixel (X + Q, Y + P).

Call Putpixel (X - Q, Y + P).

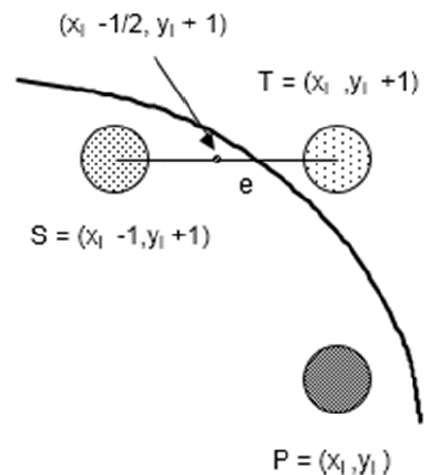
Call Putpixel (X + Q, Y - P).

Call Putpixel (X - Q, Y - P).

The midpoint circle algorithm is a commonly used algorithm for drawing circles in computer graphics. It works by dividing the circle into eight octants and using symmetry properties to generate all the points on the circle.

Here are the steps involved in the midpoint circle algorithm:

1. Define the center point of the circle (xc, yc) and the radius (r).
2. Set the initial values of x and y as x = 0, y = r.
3. Calculate the initial value of the decision parameter as d = 1 - r.
4. At each step, calculate the next point along the circle using the current values of x and y:
5. If the decision parameter d is less than or equal to 0, the next point is (x+1, y) and the new value of d is d = d + 2x + 3.
6. If the decision parameter d is greater than 0, the next point is (x+1, y-1) and the new value of d is d = d + 2x - 2y + 5.
7. Update the values of x and y based on the calculated point.
8. Repeat steps 4 and 5 until x is greater than or equal to y.
9. Use symmetry properties to generate the points in the other seven octants of the circle.



The midpoint circle algorithm is efficient and easy to implement, making it a popular choice for generating circles in computer graphics applications.

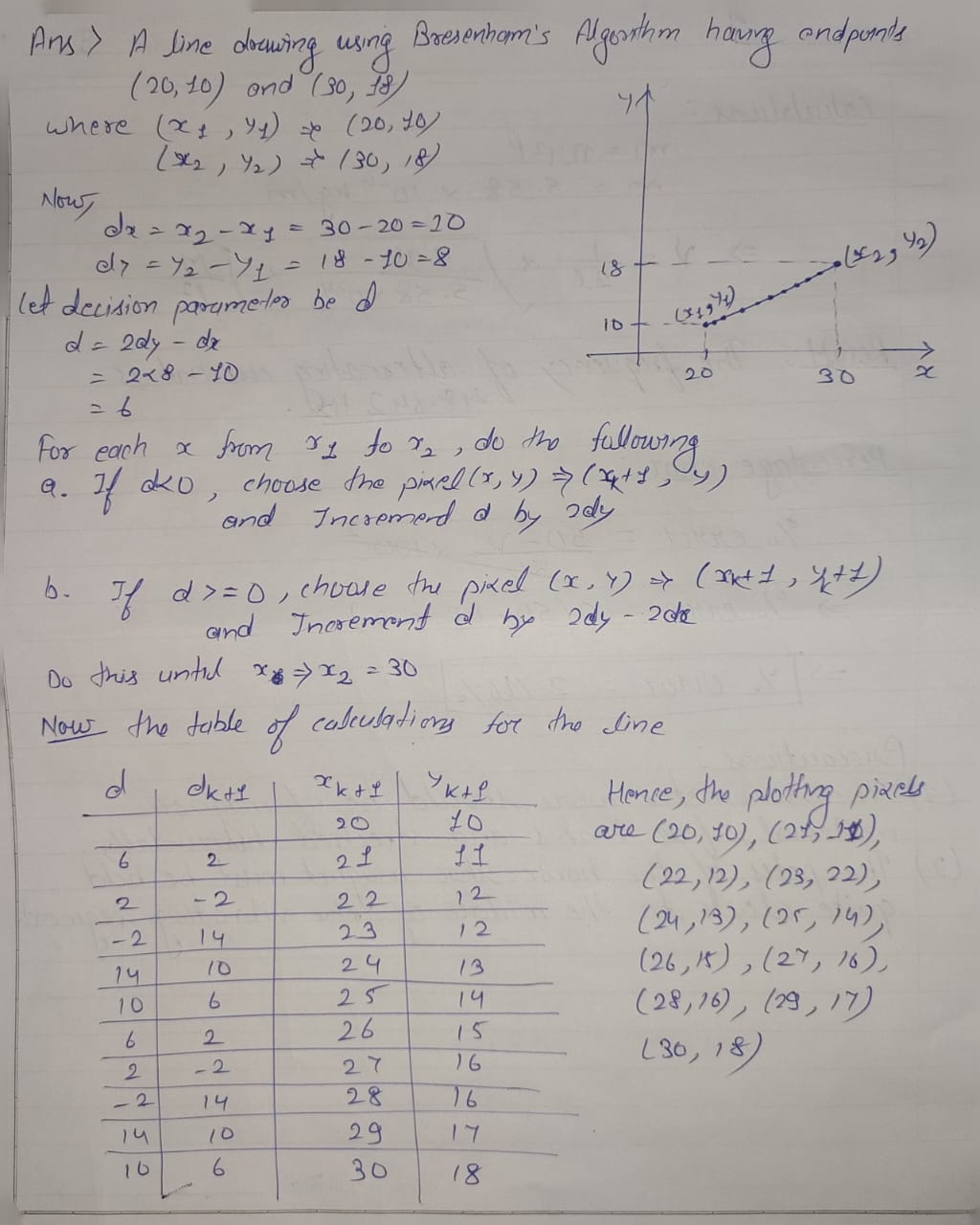
1. **Discuss the rendering pipeline in OpenGL.**

**Ans-** The rendering pipeline in OpenGL is a series of stages that transforms a set of 3D objects into a 2D image on a computer screen. The pipeline consists of several stages that are executed in a specific order, each stage transforming the input data to the next stage. Here are the stages of the rendering pipeline in OpenGL:

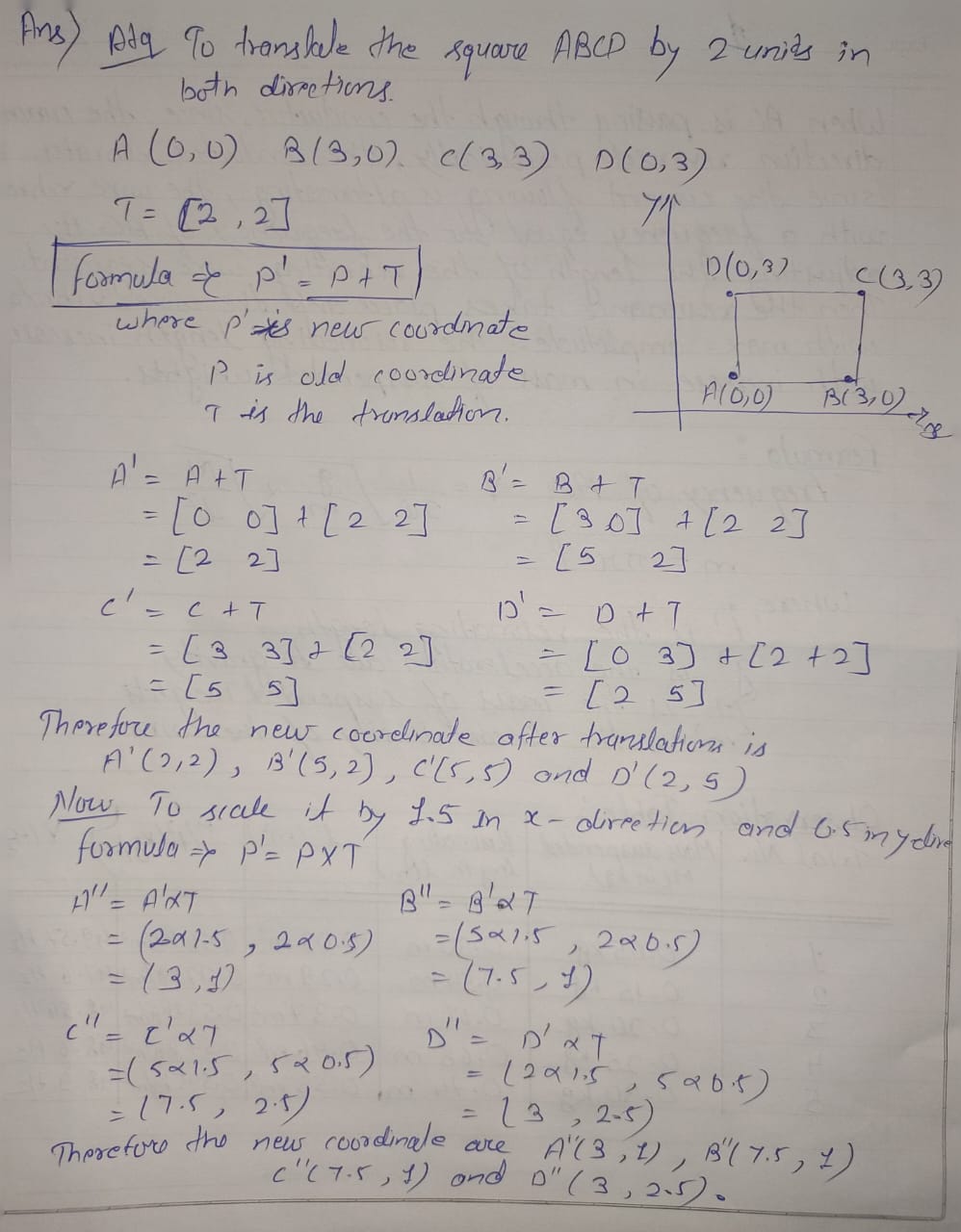
1. **Application stage:** In this stage, the 3D objects and their properties, such as position, color, and texture coordinates, are created and defined by the application code.
2. **Geometry stage:** In this stage, the 3D objects are transformed from their original coordinates to a common coordinate system, known as the world space. The objects are then further transformed to the camera space or view space, where they are visible to the camera. This stage also involves culling, clipping, and sorting of the objects.
3. **Rasterization stage:** In this stage, the 3D objects are projected onto a 2D screen space, and their shapes are approximated by a set of pixels, which are colored according to the object's properties, such as texture and lighting.
4. **Fragment stage:** In this stage, the pixels generated by the rasterization stage are processed to produce the final color of each pixel, based on the lighting, texture, and other effects applied to the objects.
5. **Output stage:** In this stage, the final image is displayed on the screen.

The rendering pipeline in OpenGL is highly configurable, allowing developers to customize each stage to achieve specific visual effects. For example, developers can add or modify shaders to customize the lighting, apply textures, and create other visual effects. Additionally, modern versions of OpenGL support parallel processing, which can significantly speed up the rendering pipeline.

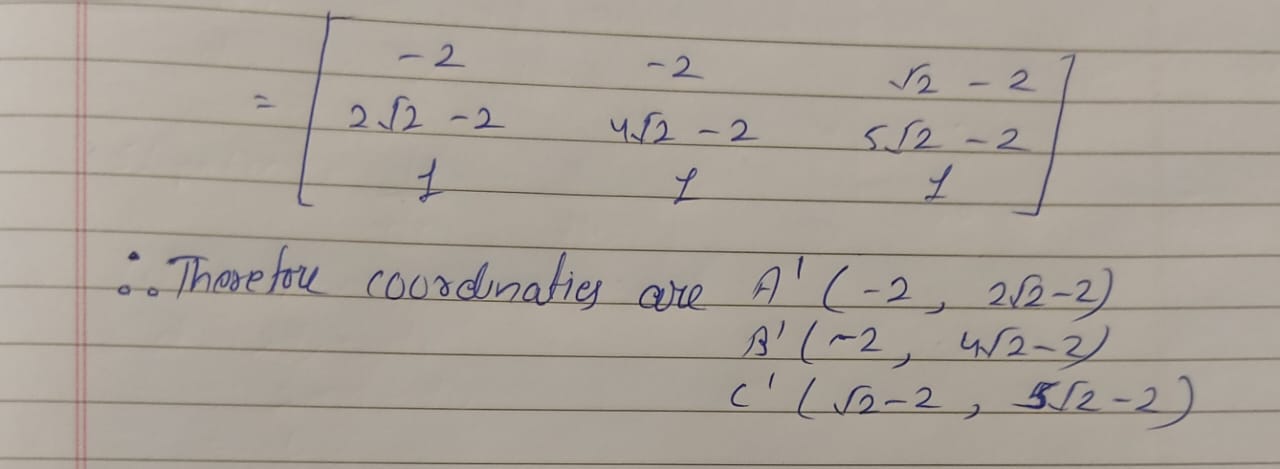
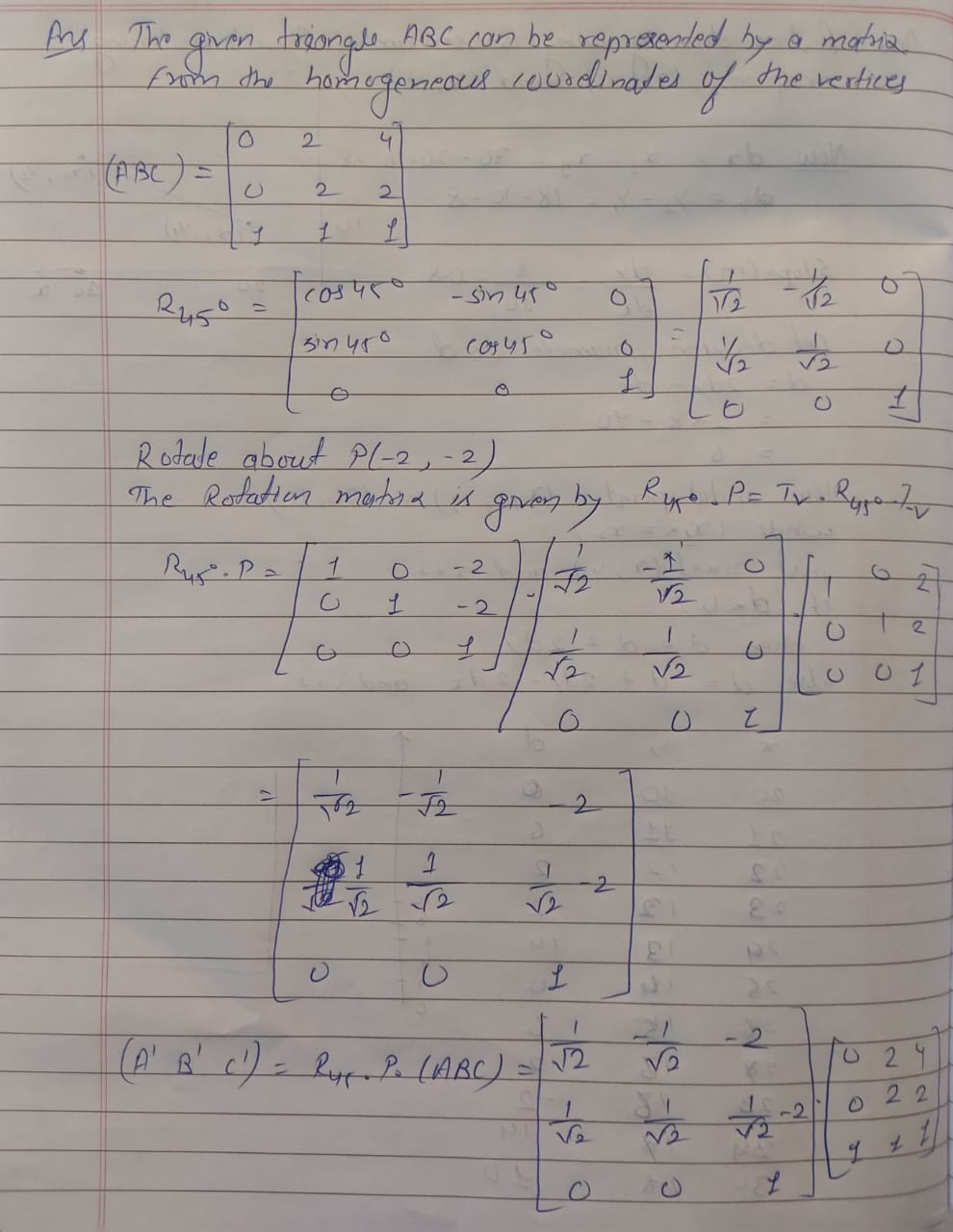
1. **Draw a line using Bresenham Algorithm having endpoints (20,10) and (30,18).[Show calculation steps]**



1. **Translate the square ABCD whose co-ordinates are A(0,0), B(3,0), C(3,3), D(0,3) by 2 units in both directions and then scale it by 1.5 in x-direction and 0.5 in y-direction.**

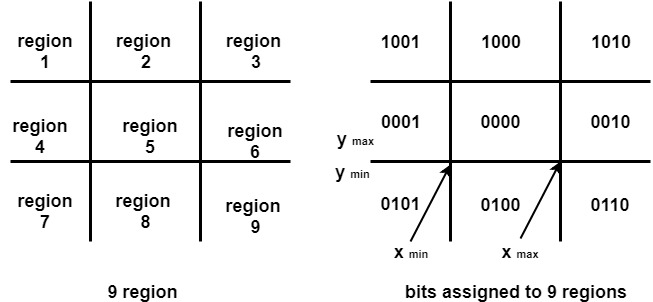
****

1. **Rotate triangle having co-ordinates A(0,0), B(2,2), C(4,2) about (a) Origin (b) P(-2,-2) by an angle of 45 degrees.**

****

1. **Explain working of Cohen Sutherland Algorithm.**

Cohen-Sutherland algorithm is a line clipping algorithm used to clip a line against a rectangular viewing window or viewport. The algorithm divides the viewport into 9 regions using 4 lines, which form a rectangular area known as the clipping window.



The algorithm uses binary codes to represent the regions of the line endpoints and the clipping window. These codes are known as the outcodes and are defined as follows:

Top = 1000 Bottom = 0100 Right = 0010 Left = 0001

For each endpoint of the line, the algorithm computes its outcode by testing its position with respect to the clipping window. If the endpoint is inside the clipping window, its outcode is 0000. If it is outside the clipping window, its outcode is determined by the region in which it lies. For example, if the endpoint is above the top edge of the clipping window, its outcode will be 0001.

The algorithm then checks if both endpoints of the line lie inside the clipping window (i.e., both outcodes are 0000). If this is the case, the line is visible and is drawn in its entirety.

If one or both endpoints lie outside the clipping window, the algorithm computes the intersection points of the line with the clipping window. To do this, it uses the slope of the line to determine which edge(s) of the clipping window the line intersects. If the slope of the line is positive, it will intersect the left and/or right edges of the clipping window. If the slope is negative, it will intersect the top and/or bottom edges.

The algorithm then updates the outcodes of the endpoints based on their new positions. If an endpoint is moved outside the clipping window, its outcode is updated accordingly. The algorithm then repeats the above steps until both endpoints are inside the clipping window or the line is determined to be completely outside the clipping window (i.e., both outcodes have a common bit set).

Once the line has been clipped, the visible portion of the line is drawn using any line-drawing algorithm, such as Bresenham's algorithm.

Overall, the Cohen-Sutherland algorithm is a simple and efficient way to clip lines against a rectangular viewport, and it forms the basis for more advanced clipping algorithms such as the Sutherland-Hodgman algorithm.

1. **Using Sutherland Hodgman Polygon Clipping Algorithm clip the following polygon.**

Ans- 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.

