Q1)

a) Define the terms: color depth, scan conversion, refresh rate. pixel.

Color depth: (number of colors) is determined by the number of bits assigned to hold color value.

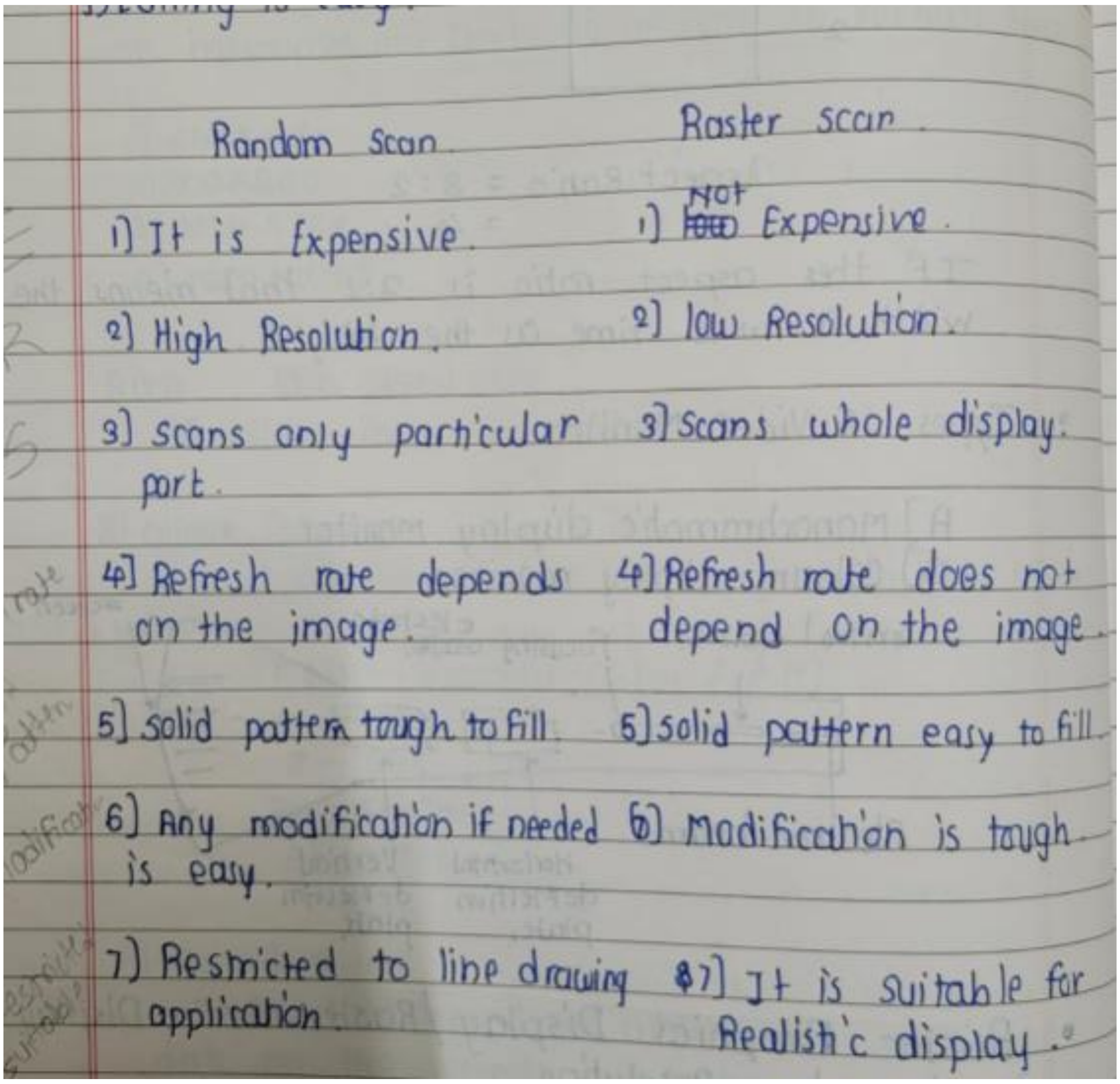
1 bit-2 colors(black n white)
4 bits-16 colors
8 bits-256 colors
16 bits -32 thousand colors
24 bits- 16 million (high color)
32 bits- latest(true color)

Rasterization: or Scan Conversion is defined as the process of representing continuous graphic object as a collection of discrete pixels.

Refresh rate: is nothing but the speed by which a particular dot on the screen is getting printed. The refresh rate is the number of times a display's image is repainted or refreshed per second. The refresh rate is expressed in hertz so a refresh rate of 75 means the image is refreshed 75 times in a second.

Pixel: The pixel is the smallest addressable screen element. It is the smallest piece of the display screen which we can control.

b) Differentiate between Raster Scan and Random Scan.



Random scan.	Raster scan.
1) It is Expensive.	1) Not ~~[illegible]~~ Expensive.
2) High Resolution.	2) low Resolution.
3) scans only particular part.	3) Scans whole display.
4] Refresh rate depends on the image.	4] Refresh rate does not depend on the image.
5] solid pattern tough to fill.	5] solid pattern easy to fill.
6] Any modification if needed is easy.	6] Modification is tough.
7) Restricted to line drawing application.	7] It is suitable for Realistic display.

Q2)

a) Define the terms: resolution, aspect ratio, frame buffer, refresh rate.

Resolution: Resolution Refers to the sharpness and clarity of an

image. It refers to the number of dots on the screen. It is expressed as a pair of numbers that give the number of dots on a horizontal line and the number such vertical lines.

Four resolutions are in common use today.

1. 640*480

2. 800*600

3. 1024*768

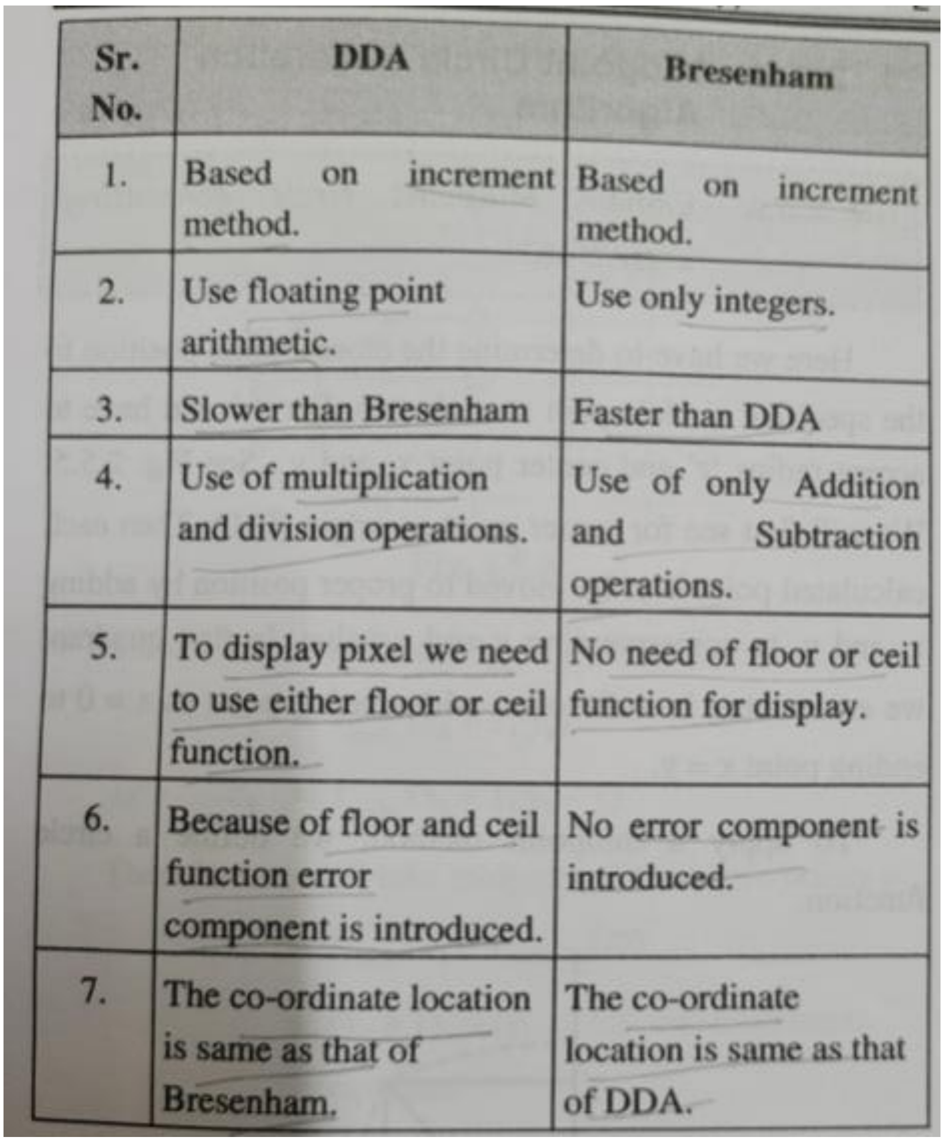
4. 1280*1024

Aspecct ratio: Aspect ratio is the ratio of width to height. In computer graphics, the relative horizontal and vertical sizes. For example, if a graphic has an aspect ratio of 2:1, it means that the width is twice as large as the height.

Frame Buffer: The frame buffer is the video memory that is used to hold or map the image that display on the screen. The portion of the memory reserved for holding the bitmapped image that is sent to the display device is called as frame buffer.

Refresh rate: is nothing but the speed by which a particular dot on the screen is getting printed. The refresh rate is the number of times a display's image is repainted or refreshed per second. The refresh rate is expressed in hertz so a refresh rate of 75 means the image is refreshed 75 times in a second.

b) Compare DDA line drawing Algorithm with Bresenhams Line drawing algorithm.



Sr. No.	DDA	Bresenham
1.	Based on increment method.	Based on increment method.
2.	Use floating point arithmetic.	Use only integers.
3.	Slower than Bresenham	Faster than DDA
4.	Use of multiplication and division operations.	Use of only Addition and Subtraction operations.
5.	To display pixel we need to use either floor or ceil function.	No need of floor or ceil function for display.
6.	Because of floor and ceil function error component is introduced.	No error component is introduced.
7.	The co-ordinate location is same as that of Bresenham.	The co-ordinate location is same as that of DDA.

Q3)

a) Write short note on “Handling Keyboard inputs with GLUT”

OpenGL Utility Toolkit (GLUT) GLUT is a library designed to provide a simple and platform-independent interface for creating graphical user interfaces

Input Handling: GLUT provides mechanisms to capture keyboard and mouse input events.

The GLUT (OpenGL Utility Toolkit) provides several functions to handle keyboard inputs easily. The functions allow you to capture when specific keys are pressed or released and then take corresponding actions.

GLUT provides two main functions for handling keyboard input:

glutKeyboardFunc()

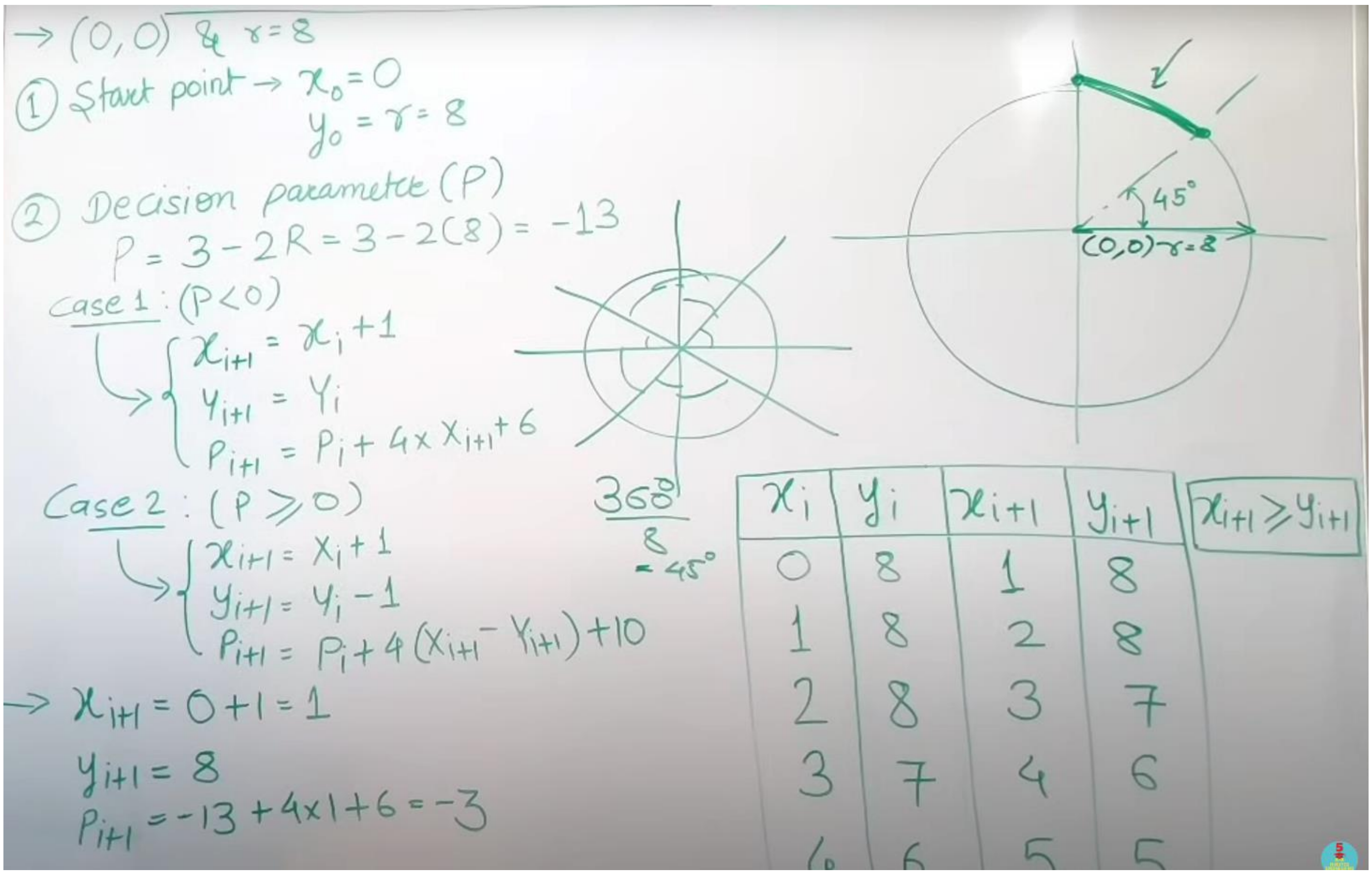
glutSpecialFunc()

The glutKeyboardFunc function is used to handle regular keyboard keys like letters, numbers, and symbols.

To handle special keys like arrow keys, function keys, or other non-ASCII keys, GLUT provides the glutSpecialFunc function.

Both glutKeyboardFunc and glutSpecialFunc callback functions receive the ASCII value of the key pressed and modifier keys (Shift, Ctrl, Alt).

b) Explain significance of error term in Bresenham’s circle drawing algorithm. Explain its mathematical derivations.



$\rightarrow (0,0)$ & $r = 8$

① Start point $\rightarrow x_0 = 0$

$y_0 = r = 8$

② Decision parameter (P)

$P = 3 - 2R = 3 - 2(8) = -13$

Case 1: $(P<0)$

$$\begin{cases} x_{i+1} = x_i + 1 \\ y_{i+1} = y_i \\ P_{i+1} = P_i + 4 \times X_{i+1} + 6 \end{cases}$$

Case 2: $(P \geq 0)$

$$\begin{cases} x_{i+1} = X_i + 1 \\ y_{i+1} = y_i - 1 \\ P_{i+1} = P_i + 4(X_{i+1} - Y_{i+1}) + 10 \end{cases}$$

$\rightarrow x_{i+1} = 0 + 1 = 1$

$y_{i+1} = 8$

$P_{i+1} = -13 + 4 \times 1 + 6 = -3$

$\frac{360}{8} = 45^\circ$

x_i	y_i	x_{i+1}	y_{i+1}	$\boxed{x_{i+1} \geq y_{i+1}}$
0	8	1	8	
1	8	2	8	
2	8	3	7	
3	7	4	6	
4	6	5	5	

Bresenham's Circle Drawing Algorithm is an efficient way to draw a circle using integer arithmetic, avoiding floating-point calculations. It is based on the idea of symmetry, where a circle’s points are plotted in one-eighth of the circle and then reflected to the other seven octants.

The key concept in Bresenham’s algorithm is the error term, which helps decide whether to increment the x-coordinate, y-coordinate, or both while stepping along the circle's perimeter.

Start at the top of the circle with the coordinates (0, r).

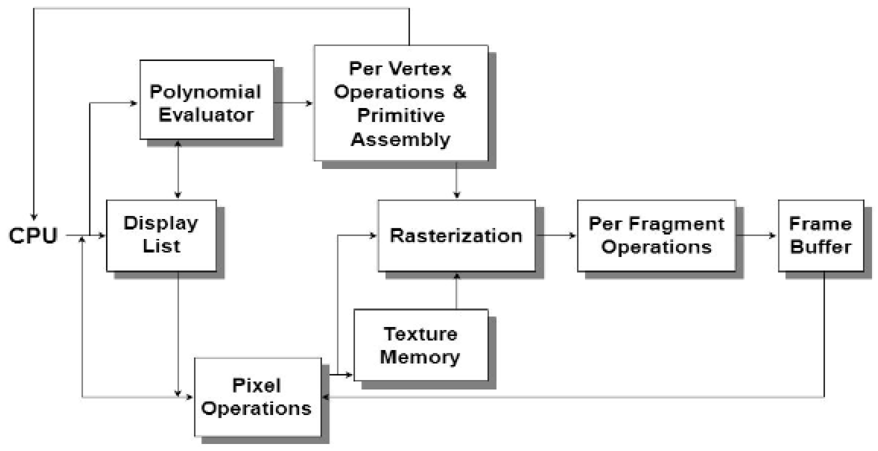
At each step, you need to decide whether to move directly right (increment x) or diagonally (increment x and decrement y). The algorithm starts by calculating the initial decision parameter

If P < 0: This means the pixel directly right (east) is closer to the true circle, so you increment x.

If P ≥ 0: The diagonal pixel (southeast) is closer, so both x is incremented, and y is decremented.

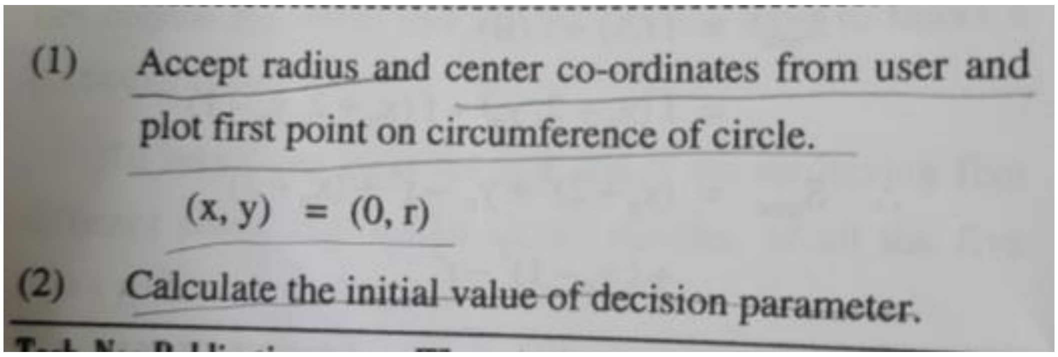
Q4)

a) Describe OpenGL architecture with block diagram in detail.



1. A display list is a sequence of OpenGL commands stored on the GPU for efficient execution.
2. The polynomial evaluator in OpenGL evaluates polynomial functions for per-vertex operations.
3. Primitive assembly is the process of combining vertices into geometric primitives
4. Rasterization is the process of converting geometric primitives into fragments (pixels) on the screen.
5. Texture memory stores images that can be applied to 3D objects for realistic surface details.
6. Per-fragment operations are computations performed on each pixel fragment during rasterization.
7. Pixel operations are computations that manipulate the final pixel values in the framebuffer.
8. The frame buffer refers to the area of memory where the final image that will be displayed on the screen is stored.

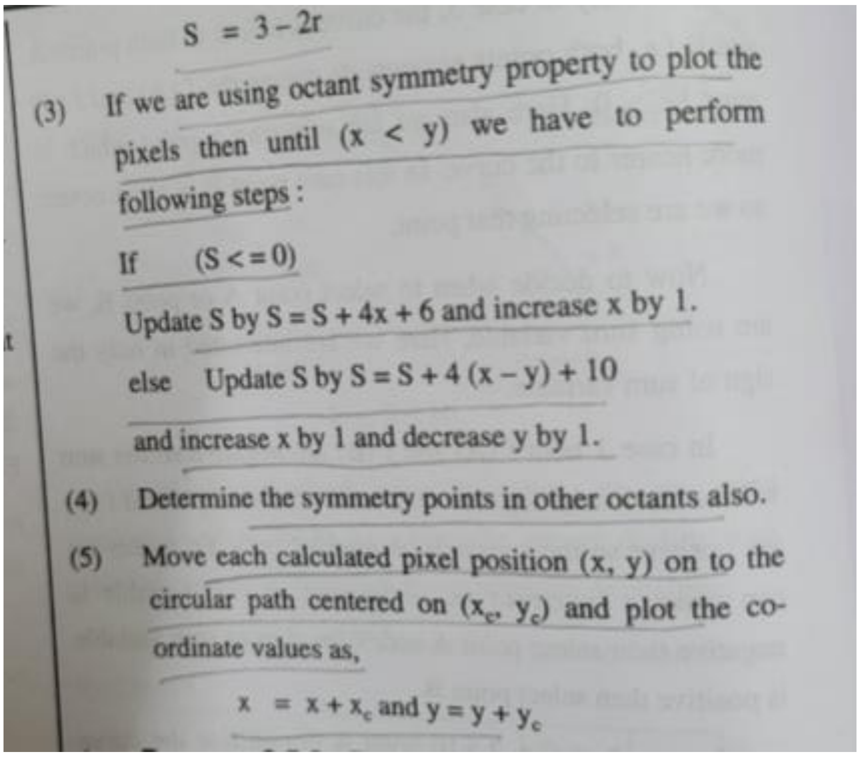
b) Explain Bresenham's circle drawing algorithm in detail.



(1) Accept radius and center co-ordinates from user and plot first point on circumference of circle.

$$(x, y) = (0, r)$$

(2) Calculate the initial value of decision parameter.



$$S = 3 - 2r$$

(3) If we are using octant symmetry property to plot the pixels then until $(x < y)$ we have to perform following steps :

If $(S <= 0)$

Update S by $S = S + 4x + 6$ and increase x by 1.

else Update S by $S = S + 4(x - y) + 10$ and increase x by 1 and decrease y by 1.

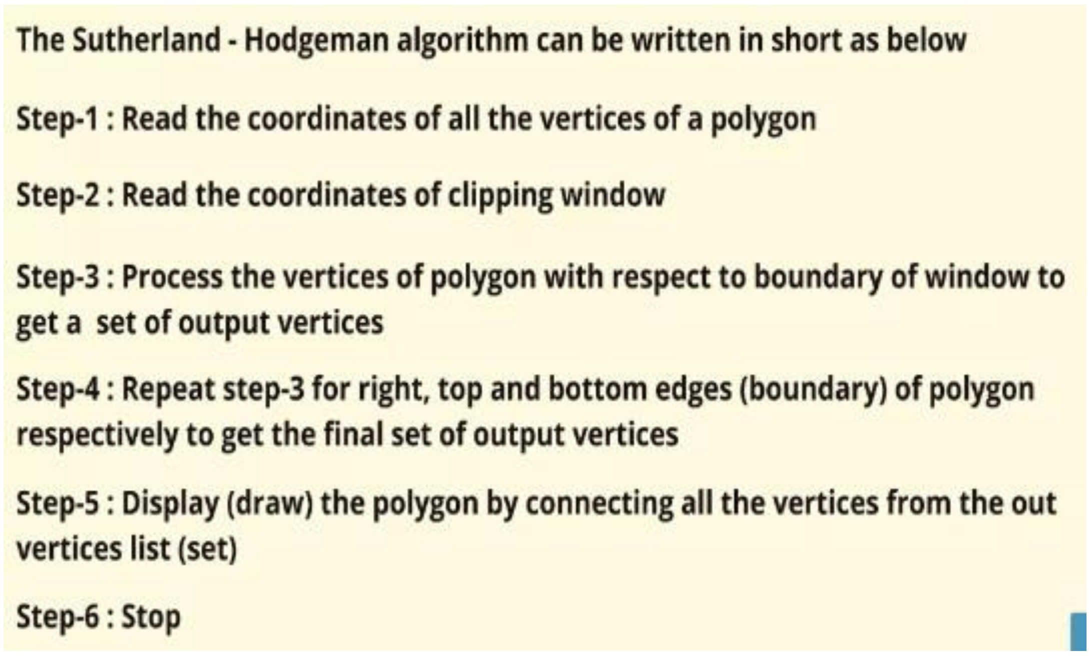
(4) Determine the symmetry points in other octants also.

(5) Move each calculated pixel position (x, y) on to the circular path centered on (x_c, y_c) and plot the co-ordinate values as,

$$x = x + x_c \text{ and } y = y + y_c$$

Q5)

a) Write and explain with example Sutherland-Hodgeman clipping algorithm



The Sutherland - Hodgeman algorithm can be written in short as below

Step-1 : Read the coordinates of all the vertices of a polygon

Step-2 : Read the coordinates of clipping window

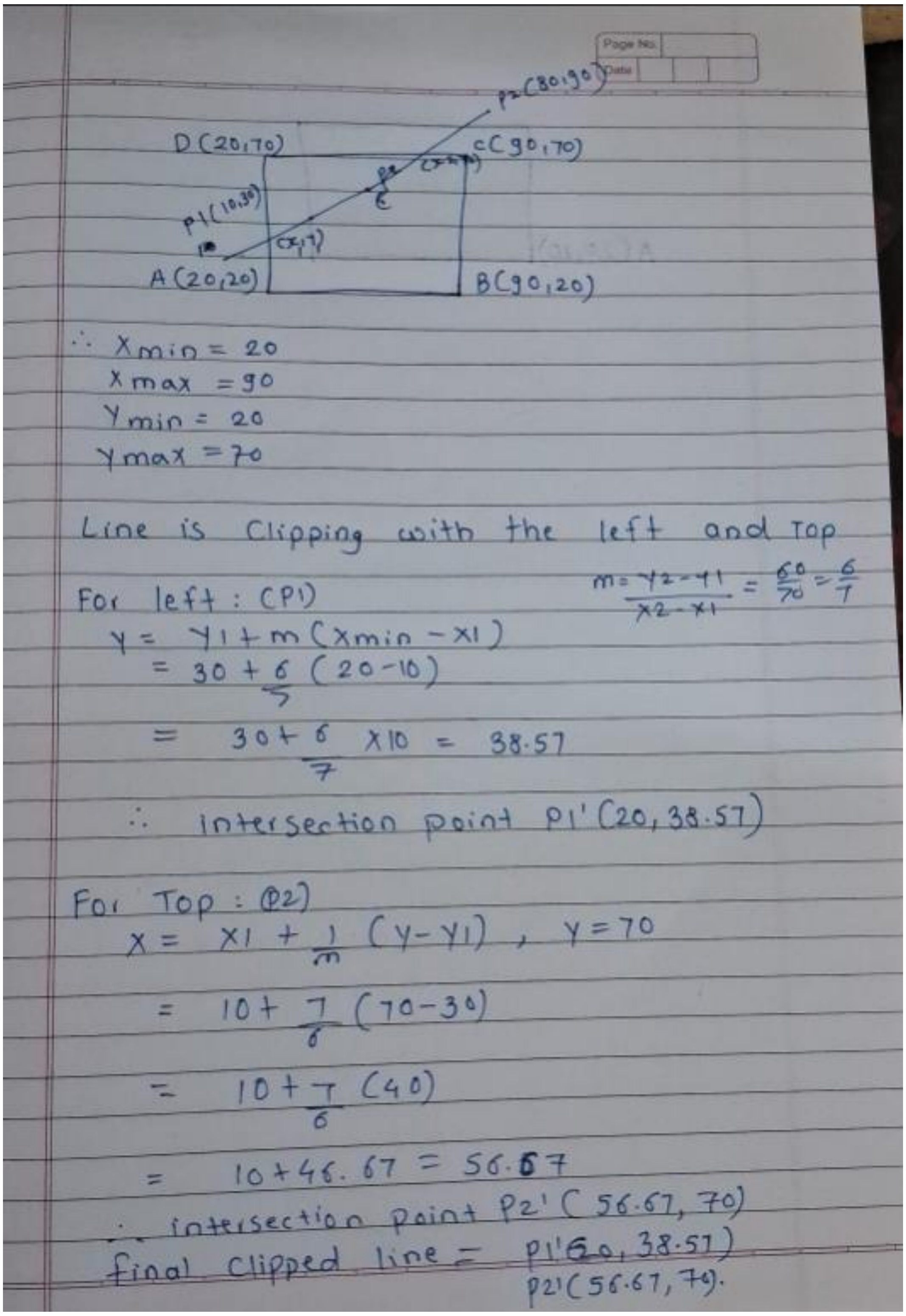
Step-3 : Process the vertices of polygon with respect to boundary of window to get a set of output vertices

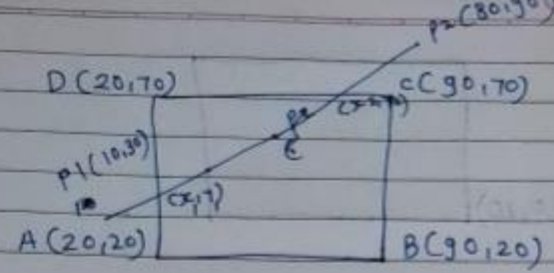
Step-4 : Repeat step-3 for right, top and bottom edges (boundary) of polygon respectively to get the final set of output vertices

Step-5 : Display (draw) the polygon by connecting all the vertices from the out vertices list (set)

Step-6 : Stop

b) Let ABCD be the rectangular window with A(20, 20), B(90, 20), C(90, 70), and D(20, 70). Find region codes for endpoints and use the Cohen-Sutherland algorithm to clip the lines : (i) P1 P2 with P1 (10, 30), P 2 (80, 90).





$\therefore$ Xmin = 20
Xmax = 90
Ymin = 20
Ymax = 70

Line is Clipping with the left and Top

$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{60}{70} = \frac{6}{7}$

For left : (P1)

$y = y_1 + m(x_{min} - x_1)$

$= 30 + \frac{6}{7}(20-10)$

$= 30 + \frac{6}{7} \times 10 = 38.57$

$\therefore$ intersection point P1' (20, 38.57)

For Top : (P2)

$x = x_1 + \frac{1}{m}(y - y_1)$, $y = 70$

$= 10 + \frac{7}{6}(70-30)$

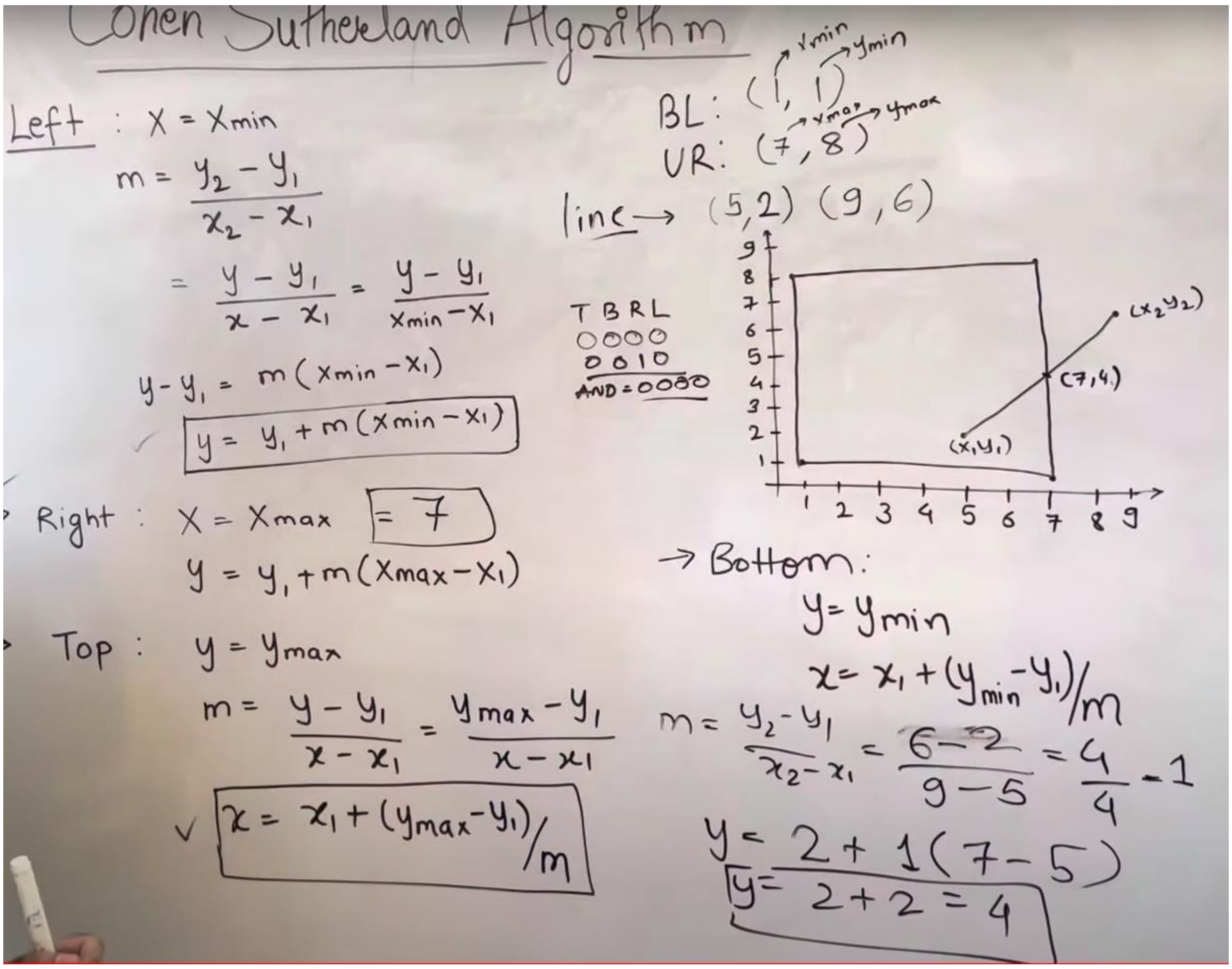
$= 10 + \frac{7}{6}(40)$

$= 10 + 46.67 = 56.67$

$\therefore$ intersection point P2' (56.67, 70)

final Clipped line = P1'(20, 38.57)
P2'(56.67, 70).

Formula reference for all sides



Cohen Sutherland Algorithm

Left : $X = X_{min}$

$m = \frac{y_2 - y_1}{x_2 - x_1}$

$= \frac{y - y_1}{x - x_1} = \frac{y - y_1}{x_{min} - x_1}$

$y - y_1 = m(x_{min} - x_1)$

✓ $\boxed{y = y_1 + m(x_{min} - x_1)}$

Right : $X = X_{max}$ $\boxed{= 7}$

$y = y_1 + m(X_{max} - x_1)$

Top : $y = y_{max}$

$m = \frac{y - y_1}{x - x_1} = \frac{y_{max} - y_1}{x - x_1}$

✓ $\boxed{x = x_1 + (y_{max} - y_1)/m}$

BL: $(\overset{x_{min}}{1}, \overset{y_{min}}{1})$

UR: $(\overset{x_{max}}{7}, \overset{y_{max}}{8})$

line → (5,2) (9,6)

TBRL
0000
0010
AND = 0000

$(x_2 y_2)$
(7,4)
(x_1, y_1)

→ Bottom:

$y = y_{min}$

$x = x_1 + (y_{min} - y_1)/m$

$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{6-2}{9-5} = \frac{4}{4} = 1$

$y = 2 + 1(7 - 5)$

$\boxed{y = 2 + 2 = 4}$

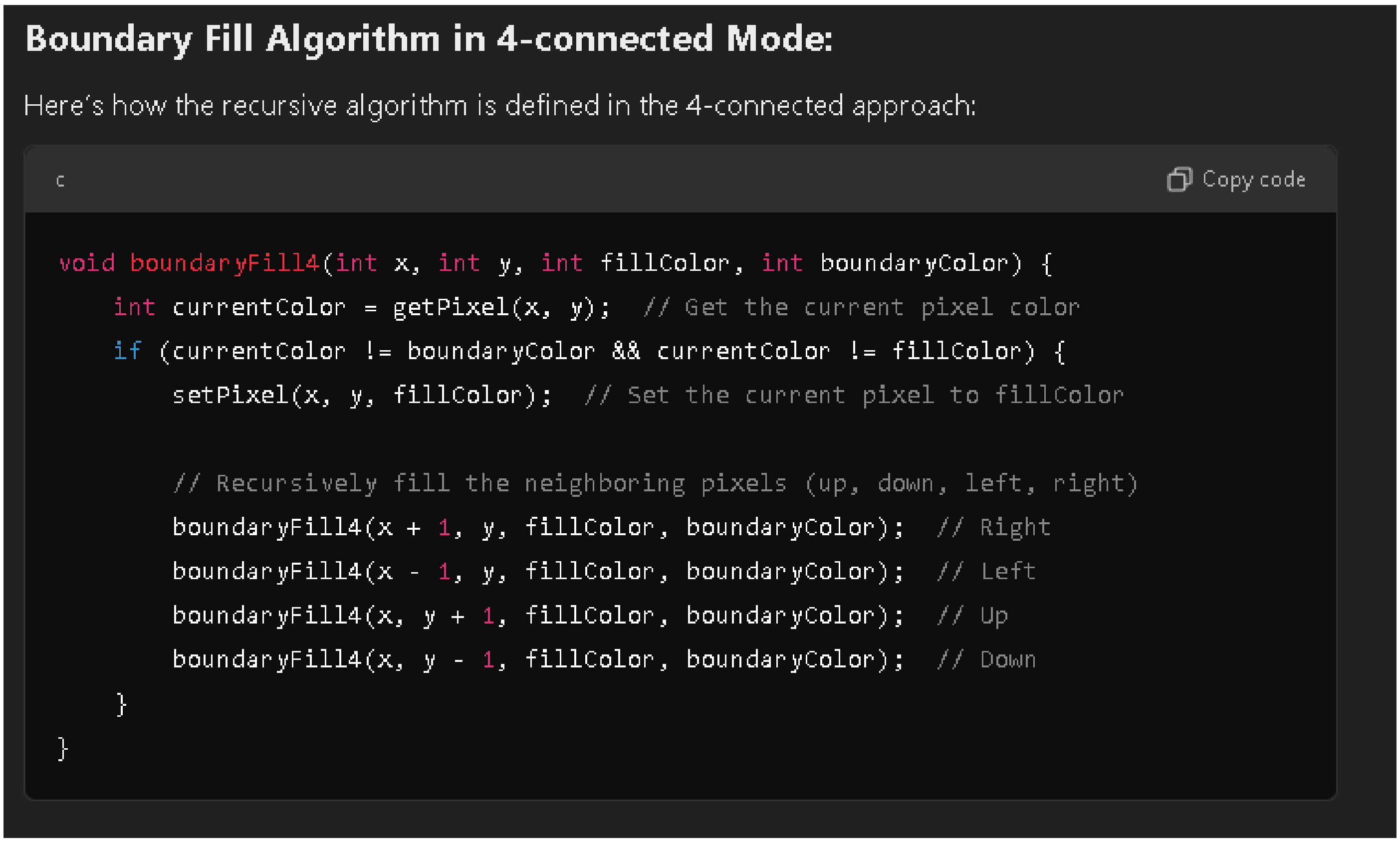
Q6)

a) Explain with an example Boundary fill Algorithm.

The Boundary Fill Algorithm is a recursive algorithm used in computer graphics to fill a connected region with a specified color

starting from a seed point inside the region. It works by filling pixels until it hits a boundary of a different color. This algorithm is particularly useful in filling polygons, areas surrounded by a defined boundary, and other irregularly shaped regions.

1. You start with a seed point, which is a pixel inside the region that needs to be filled.
2. The algorithm checks the neighboring pixels of the seed point.
3. If a neighboring pixel is not a boundary pixel and has not been filled yet, it is filled with the desired color.
4. The process repeats recursively for all neighboring pixels until the entire area is filled.



Boundary Fill Algorithm in 4-connected Mode:

Here's how the recursive algorithm is defined in the 4-connected approach:

```c
void boundaryFill4(int x, int y, int fillColor, int boundaryColor) {
    int currentColor = getPixel(x, y);  // Get the current pixel color
    if (currentColor != boundaryColor && currentColor != fillColor) {
        setPixel(x, y, fillColor);  // Set the current pixel to fillColor

        // Recursively fill the neighboring pixels (up, down, left, right)
        boundaryFill4(x + 1, y, fillColor, boundaryColor);  // Right
        boundaryFill4(x - 1, y, fillColor, boundaryColor);  // Left
        boundaryFill4(x, y + 1, fillColor, boundaryColor);  // Up
        boundaryFill4(x, y - 1, fillColor, boundaryColor);  // Down
    }
}
```

b) Clip the line PQ having coordinates P(4, 1) and Q(6, 4) against the clip window having vertices A(3, 2), B(7, 2) C(7, 6) D(3, 6). Use cohen-sutherland algo.