NAME :- KRUNAL RANK

ROLL NO :- U18CO081

CLASS :- BTECH 3RD YEAR
COMPUTER ENGINEERING

SEMESTER :- 6

EXAM :- MID SEMESTER EXAM

U18CO084

TOTAL PAGES = 6

Q1

SECTION A

Ans 1:

Ans 1: The ~~fot~~ given matrix represents the rotation of line in 3 dimensional space about x axis at an angle θ clockwise.

It can also be known as Back-X rotation.

Ans 2: The window graphics defines the points on the graphics represented in the world.
The viewport specifies the area of graphics ~~to~~ displayed on the device.

Ans 4: At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next line. This process is called retracing.

The return to the left side is known as horizontal retracing. When the last scan line is displayed, the beam returns to the first line, known as vertical retracing.

U18CO081

Q2

Ans 2.

Ans 5. DDA algorithm of drawing a line involves calculations that in turn involves floating point precession numbers.
Floating point numbers are relatively complex to ~~cal~~ calculate
This slows down the algorithm significantly.
Hence,
Bresenham's Line algorithm is used to overcome this limitation and it can also be implemented on hardware as it does not involve complex multiplication and division.

The points from which a line is to be drawn are as follows:-

(20,10) to (30,18)

As we can see here,

$P_1 = (20,10)$ $\quad \Delta y = y_2 - y_1 = 18-10 = 8$ $\quad d_1 = 16$

$P_2 = (30,18)$ $\quad \Delta x = x_2 - x_1 = 30-20 = 10$ $\quad d_2 = 16-20 = -4$

$P_0 = 2\Delta x - \Delta y = 2\times10 - 8 = 12$

P_{i-1}	x_i	y_i	P_i
12	20	10	8
8	21	11	4
4	22	12	0
0	23	13	-4
-4	24	13	12
12	25	14	8
8	26	15	4
4	27	16	0
0	28	17	-4
-4	29	17	12

P_{i-1}	x_i	y_i	P_i
12	30	18	- (Stop).

For plotting the points,

$P_0 = 2d_y - d_x = 2 \times 8 - 10 = 6$

Hence,

P_{i-1}	X_i	Y_i	P_i
6	20	10	2
2	21	11	-2
-2	22	12	

X_i	Y_i	P_i
20	10	6
21	11	2
22	12	-2
23	12	14
24	13	10
25	14	6
26	15	2
27	16	-2
28	16	14
29	17	10
30	18	6

When $P_i > 0$, $P_i += d_1 (+4)$
else, $P_i += d_2 (+12)$

U18CD084

(92)

Ans 6) Homogeneous transformation is a technique in Computer graphics that involves applying various transformations to a set of points with the help of matrix multiplication.

Here, different sets of transformation such as translation, rotation, scaling, shearing and reflection are all assigned specific matrices and when they are multiplied with the starting coordinates, the resultant coordinates are obtained.

The transformations fit into proper positions and are easier to understand, calculate and comprehend.

Homogeneous transformations help us to generalise each type of transformation into sets of matrices.

For example, for a 2D space point translation by (x', y') the starting matrix $[x_1\ y_1\ 1]$ can be multiplied with $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -x' & -y' & 1 \end{bmatrix}$

to obtain the resultant coordinates.
The last coefficient '1' in the matrix is added to preserve homogenity and matrix dimensions.

For rotation by angle θ counter clockwise, use the following matrix:-

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For shearing, $\begin{bmatrix} 1 & 0 & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} 1 & sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

(7)

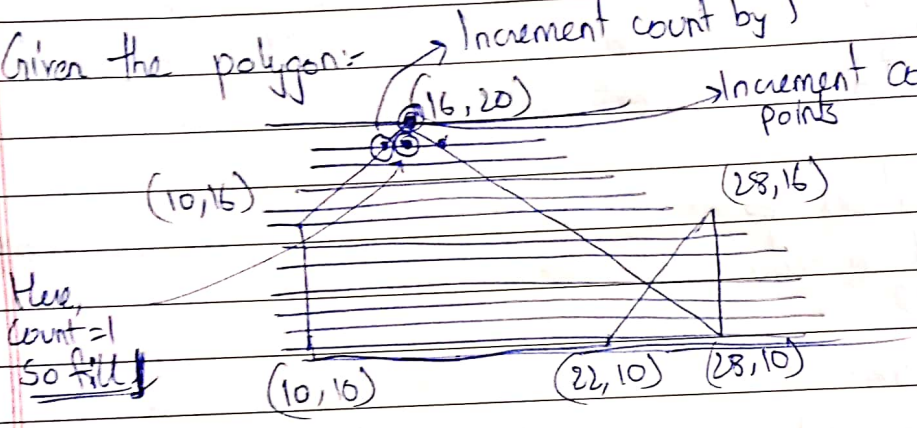
U18CC054

Q2

These matrices can be multiplied one after another to produce desired result.

Hence, it is a very convinient and reliable technique.

Ans 7. Given the polygon:-



We need to fill it using the Scan line algorithm.

Step1:- First we need to find out y_{min} & y_{max}.

Here, $y_{min} = 10$

$y_{max} = 20$

It can be done by ~~traversing~~ iterating through all sets of points and maintaining y_{min} & y_{max}.

Step2:- For ~~each~~ scan line from y_{min} to y_{max}, ~~note down~~ do the following process,

~~Starting~~

~~→ Maintain the count=0 variable.~~

Store the points where the scan line intersects the polygon lines.

For example, for scan line, $y_{20} = \{(16,20)\}$

$y_{10} = \{(10,10), (11,10), (12,10) \ldots, (22,10), (28,10)$

and so on..

U18 CO081

Q2

Step 3:- Sort all the points in respective lists. from Ymax to Ymin.

Step 4:- Sort the sides based on intersect point bases.

Step 5:- Now to start the filling for each scan line,
start from the first intersection point.
Maintain a count=0 variable.
Iterate from Xmin to Xmax.
If for any X_i, count is even, then don't fill the point, else,
fill the point.
If you encounter intersection point, increment count by 1 and
fill it with border colour.
If you encounter polygon point, increment count by 2.

Hence, the complete polygon will be filled in the same way.