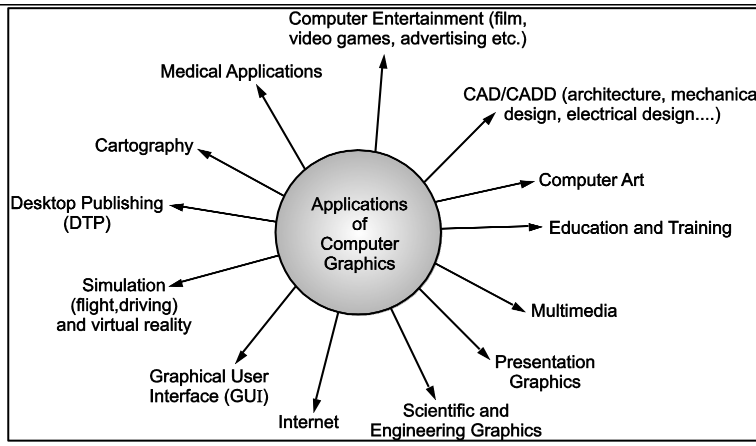
SUMMER – 19 EXAMINATION

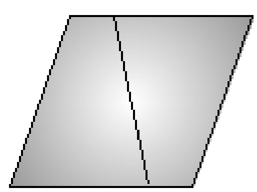
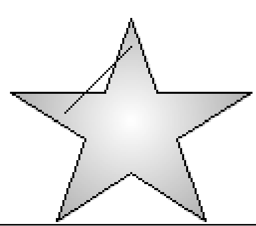
Subject Name: Computer Graphics **Model Answer** **Subject Code: 22318**

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1		**Attempt any FIVE of the following**	**10 M**
	a	**Define aspect ratio. Give one example of an aspect ratio**	**2 M**
	Ans	**Aspect ratio:** It is the ratio of the number of vertical points to the number of horizontal points necessary to produce equal length lines in both directions on the screen. **or** 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. **or** Aspect ratio is the ratio between width of an image and the height of an image. **Example:** The term is also used to describe the dimensions of a display resolution. For example, a resolution of 800x600, 1027x768, 1600x1200 has an aspect ratio of 4:3. Resolution 1280x1024 has an aspect ratio 5:4 Resolution 2160x1440, 2560x1700 has an aspect ratio 3:2	Definition-1M Example-1M
	b	**List any four applications of computer graphics.**	**2 M**

	Ans	 • DTP (Desktop Publishing) • Graphical User Interface (GUI) • Computer-Aided Design • Computer-Aided Learning (Cal) • Animations • Computer Art • Entertainment • Education and training • Image processing • Medical Applications • Presentation and Business Graphics • Simulation and Virtual Reality	Listing of four applications- 2 M
	c	**Define virtual reality. List any two advantages of virtual reality.**	**2 M**
	Ans	Virtual reality (VR) means experiencing things through our computers that don't really exist. **OR** Virtual Reality (VR) is the use of computer technology to create a simulated environment. Instead of viewing a screen in front of them, users are immersed and able to interact with 3D worlds. **Advantages:** • Virtual reality creates a realistic world • Through Virtual Reality user can experiment with an artificial environment. • Virtual Reality make the education more easily and comfort. • It enables user to explore places. • Virtual Reality has made watching more enjoyable than reading. Virtual reality widely used in video games, engineering, entertainment, education, design, films, media, medicine and many more.	Definition- 1M Any two Advantages- 1 M
	d	**List any two line drawing algorithms. Also, list two merits of any line drawing algorithm.**	**2 M**
	Ans	Line drawing algorithms:	Listing-1 M

		• Digital Differential Analyzer (DDA) algorithm • Bresenham's algorithm **Merits of DDA algorithms:** • It is the simplest algorithm and it does not require special skills for implementation. • It is a faster method for calculating pixel positions than the direct use of equation y = mx + b. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or v direction to find the pixel positions along the line path • Floating point Addition is still needed. **Merits of Bresenham's Algorithm:** • Bresenhams algorithm is faster than DDA algorithm • Bresenhams algorithm is more efficient and much accurate than DDA algorithm. • Bresenham's line algorithm is a highly efficient incremental method over DDA. • Bresenhams algorithm can draw circles and curves with much more accuracy than DDA algorithm. It produces mathematically accurate results using only integer addition, subtraction, and multiplication by 2, which can be accomplished by a simple arithmetic shift operation.	Two merits-1 M
	e	**Define convex and concave polygons.**	**2 M**
	Ans	**Convex Polygon:** It is a polygon in which if you take any two positions of polygon then all the points on the line segment joining these two points fall within the polygon itself.  **Concave Polygon:** It is a polygon in which if you take any two positions of polygon then all the points on the line segment joining these two points does not fall entirely within the polygon. 	Each 1 M
	f	**What is homogeneous co-ordinate? Why is it required?**	**2 M**
	Ans	Homogeneous coordinates are another way to represent points to simplify the way	Definition-1

in which we express affine transformations.
Normally, book-keeping would become tedious when affine transformations of the form $A\bar{p} + \vec{t}$ are composed. With homogeneous coordinates, affine transformations become matrices, and composition of transformations is as simple as matrix multiplication.

With homogeneous coordinates, a point $\bar{p}$ is augmented with a 1, to form $\hat{p} = \begin{bmatrix} \bar{p} \\ 1 \end{bmatrix}$

.

All points $(\alpha\bar{p}, \alpha)$ represent the same point $\bar{p}$ for real $\alpha \neq 0$.

OR

We have to use 3×3 transformation matrix instead of 2×2 transformation matrix. To convert a 2×2 matrix to 3×3 matrix, we have to add an extra dummy coordinate W. In this way, we can represent the point by 3 numbers instead of 2 numbers, which is called Homogenous Coordinate system.

- Homogeneous coordinates are used extensively in computer vision and graphics because they allow common operations such as translation, rotation, scaling and perspective projection to be implemented as matrix operations

3D graphics hardware can be specialized to perform matrix multiplications on 4x4 matrices.

M
Why required-1 M

g Write the transformation matrix for y-shear. **2 M**

Ans The Y-Shear can be represented in matrix from as:

$$Y_{sh} \begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$X' = X + Sh_x \cdot Y$$

$$Y' = Y$$

For matrix-2 M

2 Attempt any THREE of the following **12 M**

a Compare vector scan display and raster scan display (write any 4 points) **4M**

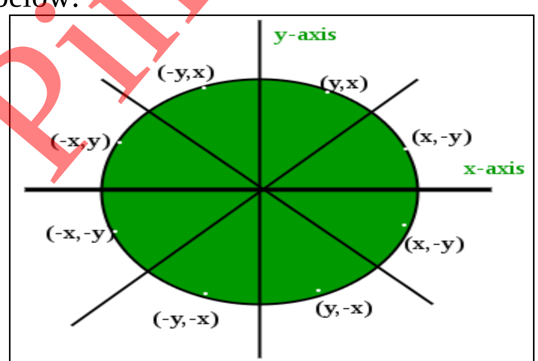
Ans

Raster	Vector
Raster graphics are composed of pixels.	Vector graphics are composed of paths.
Raster graphics are resolution dependent.	Vector graphics are resolution independent
More expensive	Less expensive.
Graphics primitives are specified in terms of their endpoints and must be scan converted into their	Scan conversion is not required

Any four point-4 M

corresponding points in the frame buffer.	
It required separate scan conversion hardware.	Scan conversion hardware is not required.
Raster display has ability to display areas filled with solid colors or patterns.	Vector display only draws lines and characters
It uses interlacing	It does not used interlacing
This displays have lower resolution	This displays have higher resolution
They occupies more space which depends on image quality.	They occupies less space
File extensions are: .bmp, .gif, .jpg, .tif	File extensions are: .pdf, .ai, .svg, .eps, .dxf

	b	**Rephrase the Bresenham's algorithm to plot 1/8th of the circle and write the algorithm required to plot the same.**	**4M**
	Ans	The key feature of circle that it is highly symmetric. So, for whole 360 degree of circle we will divide it in 8-parts each octant of 45 degree. In order to that we will use Bresenham's Circle Algorithm for calculation of the locations of the pixels in the first octant of 45 degrees. It assumes that the circle is centered on the origin. So for every pixel (x, y) it calculates, we draw a pixel in each of the 8 octants of the circle as shown below:	Rephrase-2 M Algorithm-2 M



For a pixel (x,y) all possible pixels in 8 octants.

Algorithm:

Step 1: Read the radius of circle (r).

Step 2: Set decision parameter d = 3 – 2r.

Step 3: x=0 and y=r.

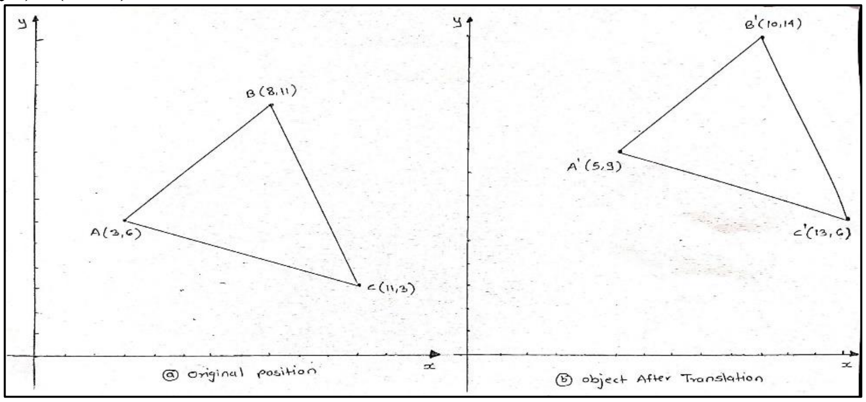
Step 4: do

{

Plot (x,y)

If(d<0) then

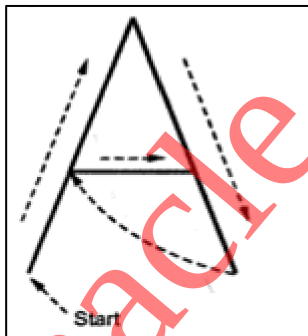
		{ d = d + 4x + 6 } Else { d=d + 4(x-y) + 10 y=y-1 } X=x-1 } While(x<y) **Step 5:** stop Plotting 8 points, each point in one octant Call Putpixel (X + h, Y + k). Call Putpixel (-X + h, Y + k). Call Putpixel (X + h, -Y + k). Call Putpixel (-X + h, -Y + k). Call Putpixel (Y + h, X + k). Call Putpixel (-Y + h, X + k). Call Putpixel (Y + h, -X - k). Call Putpixel (-Y + h,-X + k).	
	c	**Translate the polygon with co-ordinates A (3, 6), B (8, 11), & C (11, 3) by 2 units in X direction and 3 units in Y direction.**	**4M for proper solution**
	Ans	X'=x+tx Y'=y+ty tx=2 ty=3 for point A(3,6) x'=3+2=5 y'=6+3=9 for point B(8,11) x'=8+2=10 y'=11+3=14 for point C(11,3) x'=11+2=13 y'=3+3=6 A'=(x',y')=(5,9)	

		B'=(x',y')=(10,14) C'=(x',y')=(13,6)  	
	d	**Write the midpoint subdivision algorithm for line clipping.**	**4M**
	Ans	**Step 1:** Scan two end points for the line $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$. **Step 2:** Scan corners for the window as $(\omega x_1, \omega y_1)$ and $(\omega x_2, \omega y_2)$. **Step 3:** Assign the region codes for endpoints P_1 and P_2 by initializing code with 0000. Bit 1 - if $(x < \omega x_1)$ Bit 2 - if $(x > \omega x_2)$ Bit 3 - if $(y < \omega y_2)$ Bit 4 - if $(y > \omega y_1)$ **Step 4:** Check for visibility of line P_1, P_2. • If region codes for both end points are zero then the line is visible, draw it and jump to step 6. • If region codes for end points are not zero and the logical Anding operation of them is also not zero then the line is invisible, reject it and jump to step 6. • If region codes for end points does not satisfies the condition in 4 (i) and 4 (ii) then line is partly visible. **Step5:** Find midpoint of line and divide it into two equal line segments and repeat steps 3 through 5 for both subdivided line segments until you get completely visible and completely invisible line segments. **Step 6:** Exit.	Algorithm-4 M
3		**Attempt any THREE of the following**	**12 M**
	a	**State the different character generation methods. Describe any one with diagram.**	**4 M**
	Ans	**Character Generator Methods:** 1) Stroke Method 2) Bitmap Method	Listing-1 M and any one method-3 M

3) Starburst Method

1) STROKE METHOD

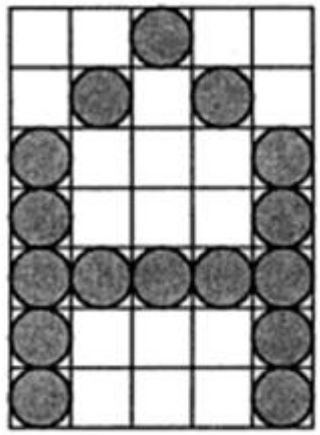
- Stroke method is based on natural method of text written by human being. In this method graph is drawing in the form of line by line.
- Line drawing algorithm DDA follows this method for line drawing.
- This method uses small line segments to generate a character. The small series of line segments are drawn like a stroke of pen to form a character.
- We can build our own stroke method character generator by calls to the line drawing algorithm. Here it is necessary to decide which line segments are needed for each character and then drawing these segments using line drawing algorithm.



2)BITMAP METHOD

- Bitmap method is a called dot-matrix method as the name suggests this method use array of bits for generating a character. These dots are the points for array whose size is fixed.
- In bit matrix method when the dots is stored in the form of array the value 1 in array represent the characters i.e. where the dots appear we represent that position with numerical value 1 and the value where dots are not present is represented by 0 in array.
- It is also called dot matrix because in this method characters are represented by an array of dots in the matrix form. It is a two dimensional array having columns and rows.

A 5x7 array is commonly used to represent characters. However 7x9 and 9x13 arrays are also used. Higher resolution devices such as inkjet printer or laser printer may use character arrays that are over 100x100.



Character A in 5 × 7 dot matrix format

3) Starbust method:

In this method a fix pattern of line segments are used to generate characters. Out of these 24 line segments, segments required to display for particular character are highlighted. This method of character generation is called starbust method because of its characteristic appearance

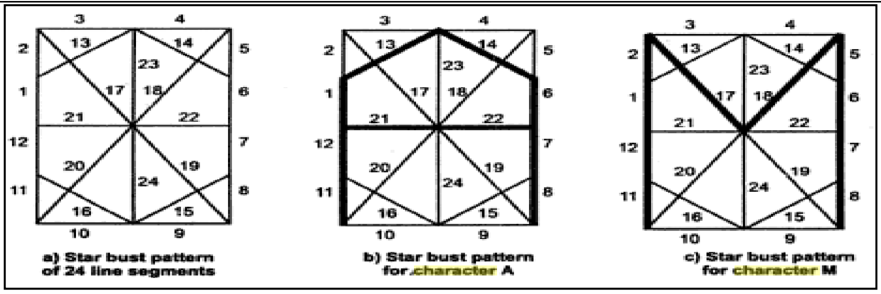
The starbust patterns for characters A and M. the patterns for particular characters are stored in the form of 24 bit code, each bit representing one line segment. The bit is set to one to highlight the line segment; otherwise it is set to zero. For example, 24-bit code for Character A is 0011 0000 0011 1100 1110 0001 and for character M is 0000 0011 0000 1100 1111 0011.

This method of character generation has some disadvantages. They are

1. The 24-bits are required to represent a character. Hence more memory is required.

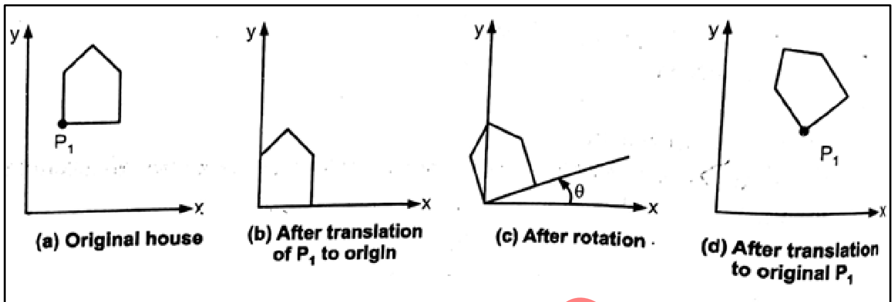
2. Requires code conversion software to display character from its 24-bit code.

3. Character quality is poor. It is worst for curve shaped characters.

		 a) Star bust pattern of 24 line segments b) Star bust pattern for character A c) Star bust pattern for character M Character A : 0011 0000 0011 1100 1110 0001 Character M:0000 0011 0000 1100 1111 0011	
	b	**Obtain a transformation matrix for rotating an object about a specified pivot point.**	**4 M**
	Ans	To do rotation of an object about any selected arbitrary point P1(x1 ,y1), following sequence of operations shall be performed. **1. Translate:** Translate an object so that arbitrary point P1 is moved to coordinate origin. **2. Rotate:** Rotate object about origin. **3. Translate:** Translate object so that arbitrary point P1 is moved back to the its original position. Note: Here to do one operation we are doing the sequence of three operations. So it is called as composite transformation or concatenation. Rotate about point P1(x1,y1). 1) Translate P1 to origin. 2) Rotate 3) Translate back to P1. Equation for this composite transformation matrix form is as follows: $P' = T(x_1, y_1) \cdot R(\theta) \cdot T(-x_1, -y_1)$ $P' = \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ Here (x1,y1) are coordinates of point P1 and hence are translation factors tx and ty; we want to move P1 to origin , x1 and y1 are x and y distances to P1and hence it is translation factor.	Proper Explanation 4 M

$$P' = \begin{bmatrix} \cos\theta & -\sin\theta & x_1(1-\cos\theta)+y_1\sin\theta \\ \sin\theta & \cos\theta & y_1(1-\cos\theta)-x_1\sin\theta \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

It is demonstrated in following figure:



c **Describe Sutherland-Hodgeman algorithm for polygon clipping.**

Ans

Explanation- 1 M and Algorithm- 3 M

- In Sutherland-Hodgeman, a polygon is clipped by processing the polygon boundary as a whole against each window edge. Clipping window must be convex.
- This could be accomplished by processing all polygon vertices against each clip rectangle boundary in turn beginning with the original set of polygon vertices, first clip the polygon against the left rectangle boundary to produce a new sequence of vertices.
- The new set of vertices could then be successively passed to a right boundary clipper, a top boundary clipper and a bottom boundary clipper.
- At each step a new set of polygon vertices us generated and passed to the next window boundary clipper. This is the logic used in Sutherland-Hodgeman algorithm.

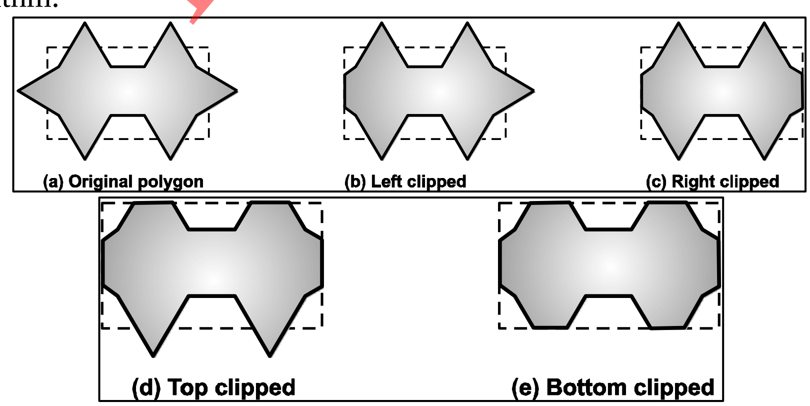


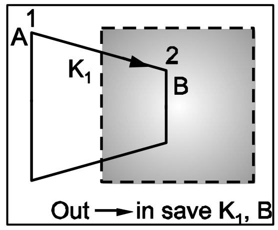
Fig. Clipping polygon against successive window boundaries

- The output of algorithm is a list of polygon vertices all of which are on the visible side of clipping plane. Such each edge of the polygon is individually compared with the clipping plane.
- This is achieved by processing two vertices of each edge of the polygon

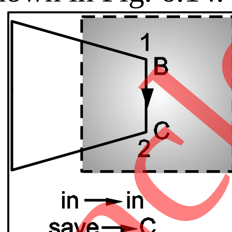
around the clipping boundary or plane.

- This results in four possible relationships between the edge and clipping plane.

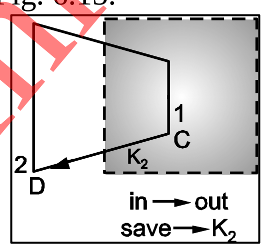
1. If first vertex of polygon edge is outside and second is inside window boundary, then intersection point of polygon edge with window boundary and second vertex are added to output vertices set as shown in Fig. 6.13.



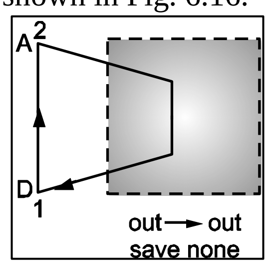
2. If both vertices of edge are inside window boundary, then add only second vertex to output set as shown in Fig. 6.14.



3. If first vertex of edge is inside and second is outside of window boundary then point of intersection of edge with window boundary is stored in output set as shown in Fig. 6.15.



4. If both vertices of edges are outside of window boundary then those vertices are rejected as shown in Fig. 6.16.



- Going through above four cases we can realize that there are two key processes in this algorithm:
  1. Determine the visibility of point or vertex (Inside – Outside Test)
  2. Determine the intersection of the polygon edge and clipping plane.
- The second key process in Sutherland-Hodgeman polygon clipping algorithm is to determine the intersection of the polygon edge and clipping plane.

		• Assume that we're clipping a polygon's edge with vertices at (x_1, y_1) and (x_2, y_2) against a clip window with vertices at (x_{min}, y_{min}) and (x_{max}, y_{max}). 1. The location (I_X, I_Y) of the intersection of the edge with the left side of the window is: (i) $I_X = x_{min}$ (ii) $I_Y = slope*(x_{min} - x_1) + y_1$, where the slope $= (y_2 - y_1)/(x_2 - x_1)$. 2. The location of the intersection of the edge with the right side of the window is: (i) $I_X = x_{max}$ (ii) $I_Y = slope*(x_{max} - x_1) + y_1$, where the slope $= (y_2 - y_1)/(x_2 - x_1)$ 3. The intersection of the polygon's edge with the top side of the window is: (i) $I_X = x_1 + (y_{max} - y_1) / slope$ (ii) $I_Y = y_{max}$ 4. Finally, the intersection of the edge with the bottom side of the window is: (i) $I_X = x_1 + (y_{min} - y_1) / slope$ (ii) $I_Y = y_{min}$ **Algorithm for Sutherland-Hodgeman Polygon Clipping:** **Step 1:** Read co-ordinates of all vertices of the polygon. **Step 2:** Read co-ordinates of the clipping window. **Step 3:** Consider the left edge of window. **Step 4:** Compare vertices of each of polygon, individually with the clipping plane. **Step 5:** Save the resulting intersections and vertices in the new list of vertices according to four possible relationships between the edge and the clipping boundary. **Step 6:** Repeat the steps 4 and 5 for remaining edges of clipping window. Each time resultant list of vertices is successively passed to process next edge of clipping window. **Step 7:** Stop.	
	d	**Given the vertices of Bezier Polygon as $P_0(1, 1)$, $P_1(2,3)$, $P_2(4,3)$, $P_3(3,1)$, determine five points on Bezier Curve.**	**4 M**

Ans

Ans :-

The equation for the Bezier Curve is given as:

$$P(u) = (1-u)^3 P_1 + 3u(1-u)^2 P_2 + 3u^2(1-u) P_3 + u^3 P_4 \quad \text{for } 0 \le u \le 1$$

where,

$P(u)$ is the point on the curve P_1, P_2, P_3, P_4

Let us take,

$$u = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$$

$$P(0) = P_1 = (1,1)$$

$$\therefore P\left(\frac{1}{4}\right) = \left(1-\frac{1}{4}\right)^3 P_1 + 3\frac{1}{4}\left(1-\frac{1}{4}\right)^2 P_2 + 3\left(\frac{1}{4}\right)^2\left(1-\frac{1}{4}\right)P_3 + \left(\frac{1}{4}\right)^3 P_4$$

$$= \frac{27}{64}(1,1) + \frac{27}{64}(2,3) + \frac{9}{64}(4,3) + \frac{1}{64}(3,1)$$

$$= \left[\frac{27}{64}\times 1 + \frac{27}{64}\times 2 + \frac{9}{64}\times 4 + \frac{1}{64}\times 3, \quad \frac{27}{64}\times 1 + \frac{27}{64}\times 3 + \frac{9}{64}\times 3 + \frac{1}{64}\times 1\right]$$

$$= \left[\frac{27}{64} + \frac{54}{64} + \frac{36}{64} + \frac{3}{64}, \quad \frac{27}{64} + \frac{81}{64} + \frac{27}{64} + \frac{1}{64}\right]$$

$$= \left[\frac{120}{64}, \frac{136}{64}\right]$$

$$= (1.875, 2.125)$$

Proper result 4 M

$$\therefore P\left(\tfrac{1}{2}\right) = \left(1-\tfrac{1}{2}\right)^3 P_1 + 3\tfrac{1}{2}\left(1-\tfrac{1}{2}\right)^2 P_2 + 3\left(\tfrac{1}{2}\right)^2\left(1-\tfrac{1}{2}\right) P_3 + \left(\tfrac{1}{2}\right)^3 P_4$$

$$= \tfrac{1}{8}(1,1) + \tfrac{3}{8}(2,3) + \tfrac{3}{8}(4,3) + \tfrac{1}{8}(3,1)$$

$$= \left[\tfrac{1}{8}\times 1 + \tfrac{3}{8}\times 2 + \tfrac{3}{8}\times 4 + \tfrac{1}{8}\times 3,\ \tfrac{1}{8}\times 1 + \tfrac{3}{8}\times 3 + \tfrac{3}{8}\times 3 + \tfrac{1}{8}\times 1\right]$$

$$= \left[\tfrac{1}{8} + \tfrac{6}{8} + \tfrac{12}{8} + \tfrac{3}{8},\ \tfrac{1}{8} + \tfrac{9}{8} + \tfrac{9}{8} + \tfrac{1}{8}\right]$$

$$= \left[\tfrac{22}{8}, \tfrac{20}{8}\right]$$

$$= (2.75, 2.5)$$

$$\therefore P\left(\tfrac{3}{4}\right) = \left(1-\tfrac{3}{4}\right)^3 P_1 + 3\tfrac{3}{4}\left(1-\tfrac{3}{4}\right)^2 P_2 + 3\left(\tfrac{3}{4}\right)^2\left(1-\tfrac{3}{4}\right) P_3 + \left(\tfrac{3}{4}\right)^3 P_4$$

$$= \tfrac{1}{64} P_1 + \tfrac{9}{64} P_2 + \tfrac{27}{64} P_3 + \tfrac{27}{64} P_4$$

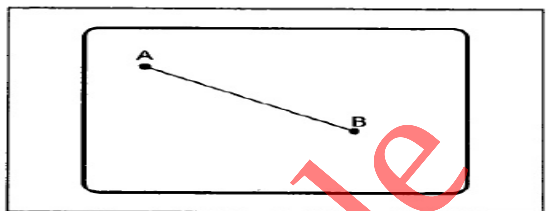
$$= \tfrac{1}{64}(1,1) + \tfrac{9}{64}(2,3) + \tfrac{27}{64}(4,3) + \tfrac{27}{64}(3,1)$$

$$= \left[\tfrac{1}{64}\times 1 + \tfrac{9}{64}\times 2 + \tfrac{27}{64}\times 4 + \tfrac{27}{64}\times 3,\ \tfrac{1}{64}\times 1 + \tfrac{9}{64}\times 3 + \tfrac{27}{64}\times 3 + \tfrac{27}{64}\times 1\right]$$

$$= \left[\tfrac{1}{64} + \tfrac{18}{64} + \tfrac{108}{64} + \tfrac{81}{64},\ \tfrac{1}{64} + \tfrac{27}{64} + \tfrac{81}{64} + \tfrac{27}{64}\right]$$

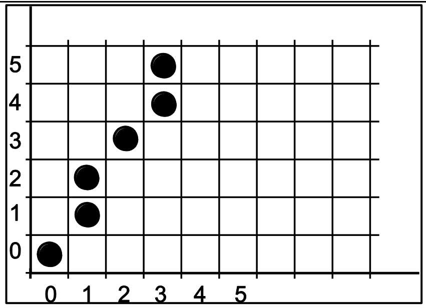
$$= \left[\tfrac{208}{64}, \tfrac{136}{64}\right] = (3.25, 2.125)$$

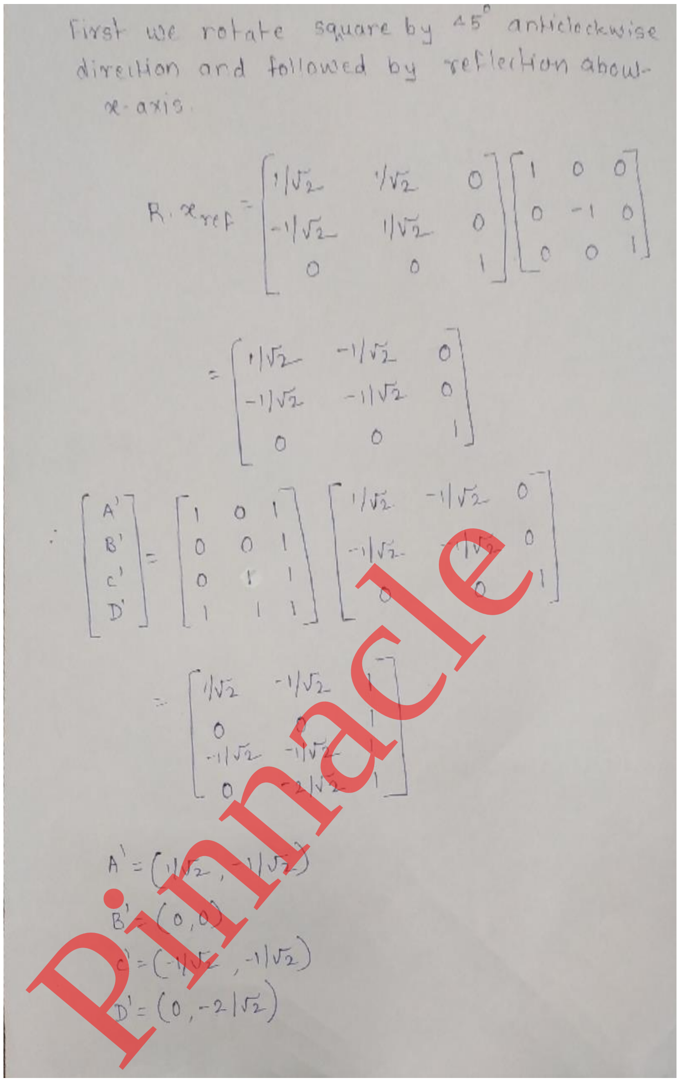
$$P(1) = P_3 = (3,1)$$

4		**Attempt any THREE of the following**	**12 M**
	a	**Describe the vector scan display techniques with neat diagram.**	**4 M**
	Ans	• A pen plotter operates in a similar way and is an example of a random-scan, hard-copy device. • When operated as a random-scan display unit, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn. • Random scan monitors draw a picture one line at a time and for this reason are also referred to as vector displays (or stroke-writing or calligraphic displays).  • Here the electron gun of a CRT illuminate's points and / or straight lines in any order. If we want a line connecting point A with point B on vector graphics display, we simply drive the beam reflection circuitry, which will cause beam to go directly from point A to point B. • Refresh rate on a random-scan system depends on the number of lines to be displayed. • Picture definition stored as a set of line drawing commands in an area of memory called "***refresh display file***" or also called as ***display list*** or ***display program*** or ***refresh buffer***. • To display a given picture, the system cycles through the set of commands in the display file, drawing each component line by line in turn. After all line drawing commands have been processed, the system cycles back to the first line drawing command in the list. And repeats the procedure of scan, display and retrace. • This displays to draw all the component lines of picture 30 to 60 frames/second • Random scan system is designed for line drawing applications; hence cannot display realistic shaded scenes. • Vector displays produces smooth line drawings but raster produces jagged lines that are plotted points • Random scan suitable for applications like engineering and scientific drawings • Graphics patterns are displayed by directing the electron beam along the component lines of the picture • A scene is then drawn one line at a time by positioning the beam to fill in the line between specified end points.	Explanation 3 M Diagram 1 M

	b	**Consider the line from (0,0) to (4,6). Use the simple DDA algorithm to rasterize this line.**	**4 M**
	Ans	Evaluating steps 1 to 5 in the DDA algorithm we have, $X_1 = 0, Y_1 = 0$ $X_2 = 4, Y_2 = 6$ Length = $\|Y_2 - Y_1\| = 6$ $\Delta X = \|X_2 - X_1\| / \text{Length} = 4/6$ $\Delta Y = \|Y_2 - Y_1\| / \text{Length} = 6/6 = 1$ Initial value for, $X = 0 + 0.5 \times (4/6) = 0.5$ $Y = 0 + 0.5 \times (1) = 0.5$ Plot integer now: 1. Plot (0,0), $x=x+\Delta X=0.5+4/6=1.167$, $y=y+\Delta Y=0.5+1=1.5$ 2.Plot (1,1), $x=x+\Delta X=1.167+4/6=1.833$ $y=y+\Delta Y=1.5+1=2.5$ 3.Plot (1,2), $x=x+\Delta X=1.833+4/6=2.5$ $y=y+\Delta Y=2.5+1=3.5$ 4.Plot (2,3), $x=x+\Delta X=2.5+4/6=3.167$ $y=y+\Delta Y=3.5+1=4.5$ 5.Plot (3,4), $x=x+\Delta X=3.167+4/6=3.833$ $y=y+\Delta Y=4.5+1=5.5$ 6.Plot (3,5), $x=x+\Delta X=3.833+4/6=4.5$ $y=y+\Delta Y=5.5+1=6.5$ Tabulating the results of each iteration in the step 7 we get,	Proper result 4 M

i	Plot	x	y
		0.5	0.5
1	(0,0)	1.167	1.5
2	(1,1)	1.833	2.5
3	(1,2)	2.5	3.5
4	(2,3)	3.167	4.5
5	(3,4)	3.833	5.5
6	(3,5)	4.5	6.5

		 **Fig. 2.2** • The results are plotted as shown in the Fig. 2.2. It shows that the rasterized line lies to both sides of the actual line, i.e. the algorithm is orientation dependent.	
	c	**Consider a square A(1,0), B(0,0), C(0,1), D(1,1). Rotate the square by 45^0 anti-clockwise direction followed by reflection about X-axis.**	**4 M**
	Ans	Given, A(1,0) B(0,0) C(0,1) D(1,1) $R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ Here, $\theta = 45^\circ$ $R = \begin{bmatrix} \cos 45 & \sin 45 & 0 \\ -\sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $= \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$ Matrix Reflection about x-axis :- $x_{ref} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Rotation + Reflection Matrix 1 M final Result= 3 M



First we rotate square by 45° anticlockwise direction and followed by reflection about x-axis.

$$R \cdot x_{ref} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

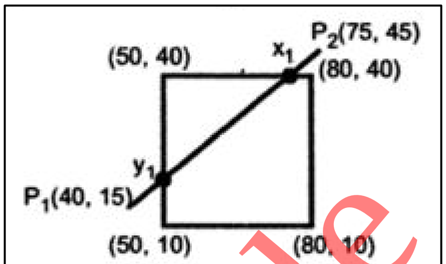
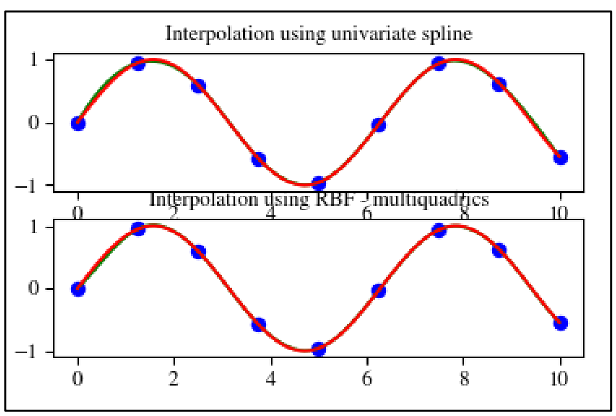
$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ -1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ -1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 1 \\ 0 & 0 & 1 \\ -1/\sqrt{2} & -1/\sqrt{2} & 1 \\ 0 & -2/\sqrt{2} & 1 \end{bmatrix}$$

$A' = (1/\sqrt{2}, -1/\sqrt{2})$
$B' = (0, 0)$
$C' = (-1/\sqrt{2}, -1/\sqrt{2})$
$D' = (0, -2/\sqrt{2})$

	d	**Use Cohen-Sutherland outcode algorithm to clip line PI (40, 15) -- P2 (75. 45) against a window A (50, 10), B (80, 10). C(80, 40) & D(50,40).**	**4 M**
	Ans	P1 (40, 15) - P2 (75, 45) Wxi = 50 Wy2 = 40 Wx2 = 80 Wy2 = 10 Point Endcode ANDing P1 0001 0000 (Partially visible) P2 0000 $y_1 = m(x_L – x) + y = \frac{6}{7}(50-40)+15$ $m = \frac{45-15}{75-40}$	Proper result 4 M

		$= 23.57$ $x_1 = \frac{1}{m}(y_T - y) + x = \frac{7}{6}(40\text{-}50)+40 = 69.16$ $y_2 = m(x_R - x) + y = \frac{6}{7}(80\text{-}40)+15 = 49.28$ $x_2 = \frac{1}{m}(y_B - y) + x = \frac{7}{6}(10\text{-}15)+40 = 34.16$ Hence:  	
	e	**What is interpolation? Describe the Lagrangian Interpolation method.**	**4 M**
	Ans	Specify a spline curve by giving a set of coordinate positions, called control points, which indicates the general shape of the curve These, control points are then fitted with piecewise continuous parametric polynomial functions in one of two ways. When polynomial sections are fitted so that the curve passes through each control point, the resulting curve is said to interpolate the set of control points. On the other hand, when the polynomials are fitted to the general control-point path without necessarily passing through any control point, the resulting curve is said to approximate the set of control points interpolation curves are commonly used to digitize drawings or to specify animation paths. Approximation curves are primarily used as design tools to structure object surfaces an approximation spline surface credited for a design application. Straight lines connect the control-point positions above the surface  	Definition- 1 M Description of Lagrangian method- 3 M

Lagrangian Interpolation Method:

Suppose we want a polynomial curve that will pass through n sample points - $(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)$, the function can be constructed as the sum of terms, one term for each sample point.

a. Blending Function :

$$f_x(u) = \sum_{i=1}^{n} x_i B_i(u)$$

$$f_y(u) = \sum_{i=1}^{n} y_i B_i(u)$$

$$f_z(u) = \sum_{i=1}^{n} z_i B_i(u)$$

The function $B_i(u)$ is called as a blending function. For each value of u, the blending function determines which i^{th} sample point affects the position of the curve.

The function $B_i(u)$ tells how hard the i^{th} sample point is pulling it for some value of u, $B_i(u) = 1$ and for each $j \neq i$, $B_j(u) = 0$, then i^{th} sample point has complete control of the curve. The curve will pass through i^{th} sample point. Create a blending function for which the sample points (x_1, y_1, z_1) has complete control when $u = -1$, the third when $u = 1$ and so on. Therefore, we require a blending function.

$B_1(u) = 1$ at $u = -1$

and $B_1(u) = 0$ at $u = 0, 1, 2, 3, \ldots, n-2$

An expression is 0 at $u (u-1)(u-2) \ldots [u-(n-2)]$

At $u = -1$, it is $(-1)(-2)(-3) \ldots (1-n)$

So dividing by above constant, it gives 1 at $u = -1$

Therefore

$$B_1(u) = \frac{u(u-1)(u-2)\ldots[(u-(n-2)]}{(-1)(-2)(-3)\ldots(1-n)}$$

The i^{th} blending function can be constructed in the same way to be 1 at $u = i - 2$ and 0 at other integers.

$$\therefore B_i(u) = \frac{(u+1)(u)(u-1)\ldots[u-(i-3)][u-(i-1)]\ldots[u-(i-2)]}{(i-1)(i-2)(i-3)\ldots(1)(-1)\ldots(i-n)}$$

The curve which is approximated using above equation is called **Lagrange Interpolation.**

5		**Attempt any TWO of the following :**	**12 M**
	a	**Consider the line from (5,5) to (13,9). Use the Bresenham's algorithm to rasterize the line.**	**6 M**
	Ans	**Bresenham Line Drawing Calculator** By putting x1,x2 and y1,y2 Value it Show The Result In Step By Step order, and Result Brief Calculation Which Is Calculated by Bresenham Line Drawing Algorithm. Bresenham Line Drawing Algorithm display result in tables. Starting Points is x1,y1 and Ending points is x2,y2.	Remark: Preliminary Calculations 2 M; Step wise plot 4 M

Preliminary Calculations:

x1 = 5 | y1 = 5 | & | x2 = 13 | y2 = 9

Calculation	**Result**

dx = abs(x1 - x2)	8 = abs(5 - 13)
dy = abs(y1 - y2)	4 = abs(5 - 9)
p = 2 * (dy - dx)	-8 = 2 * (4 - 8)
ELSE	x = x1 | y = y1 | end = x2
	x = 5 | y = 5 | end = 13

Stepwise Plot:

STEP	while(x < end)	x = x + 1	if(p < 0) { p = p + 2 * dy } else{ p = p + 2 * (dy - dx) }	OUTPUT
1	6 < 13	6 = 5 + 1	IF 0 = -8 + 2 * 4	x = 6 | y = 5
2	7 < 13	7 = 6 + 1	ELSE -8 = 0 + 2 * (4 - 8)	x = 7 | y = 6
3	8 < 13	8 = 7 + 1	IF 0 = -8 + 2 * 4	x = 8 | y = 6
4	9 < 13	9 = 8 +	ELSE -8 = 0 + 2 * (4 -	x = 9 | y = 7

		1	8)	
5	10 < 13	10 = 9 + 1	IF 0 = -8 + 2 * 4	x = 10 | y = 7
6	11 < 13	11 = 10 + 1	ELSE -8 = 0 + 2 * (4 - 8)	x = 11 | y = 8
7	12 < 13	12 = 11 + 1	IF 0 = -8 + 2 * 4	x = 12 | y = 8
8	13 < 13	13 = 12 + 1	ELSE -8 = 0 + 2 * (4 - 8)	x = 13 | y = 9

b **Apply the shearing transformation to square with A(0,0), B(1,0), C(1,1), D(0,1) as given below.**
(i) Shear Parameter value of 0.5 relative to the line yref = -1.
(ii) Shear Parameter value of 0.5 relative to the line xref = -1.

6 M

Ans We can represent the given square ABCD, in matrix form, using homogeneous coordinates of vertices as:

$$\begin{bmatrix} A & 0 & 0 & 1 \\ B & 1 & 0 & 1 \\ C & 1 & 1 & 1 \\ D & 0 & 1 & 1 \end{bmatrix}$$

i) Here $Sh_x = 0.5$ and $y_{ref} = -1$

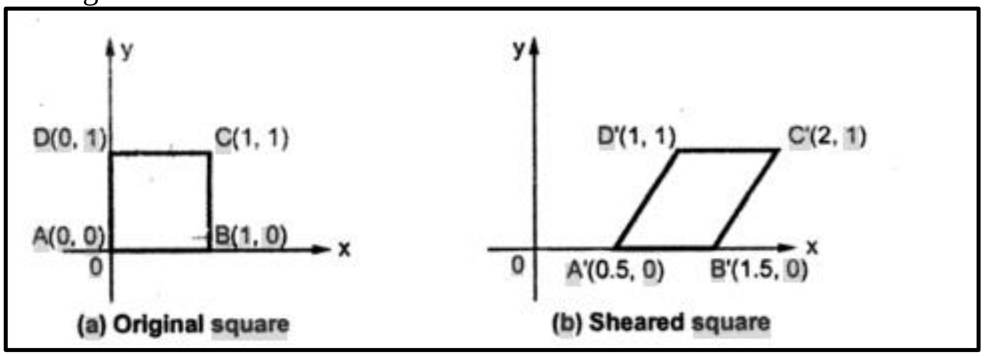
$$\begin{bmatrix} A` \\ B` \\ C` \\ D` \end{bmatrix} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ Shx & 1 & 0 \\ -Shx * yref & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & 0 & 1 \end{bmatrix}$$

Each sub problem – 3 M

$$= \begin{bmatrix} 0.5 & 0 & 1 \\ 1.5 & 0 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Shearing Transformation Result:-



(a) Original square (b) Sheared square

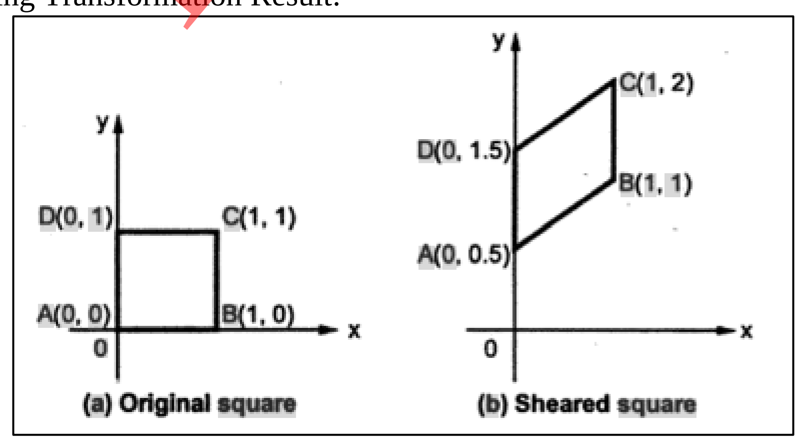
ii) Here $Sh_y = 0.5$ and $x_{ref} = -1$

$$\begin{bmatrix} A` \\ B` \\ C` \\ D` \end{bmatrix} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} * \begin{bmatrix} 1 & Shy & 0 \\ 0 & 1 & 0 \\ 0 & -Shy * xref & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0.5 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0.5 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & 1.5 & 1 \end{bmatrix}$$

Shearing Transformation Result:-



(a) Original square (b) Sheared square

	c	**Write a program in 'C' to generate Hilbert's curve.**	**6 M**
	Ans	Correct logic – 6 Marks)	

		`` #include <stdio.h> #include <stdlib.h> #include <graphics.h> #include <math.h> void move(int j,int h,int &x,int &y) { if(j==1) y-=h; else if(j==2) x+=h; else if(j==3) y+=h; else if(j==4) x-=h; lineto(x,y); } void hilbert(int r,int d,int l,int u,int i,int h,int &x,int &y) { if(i>0) { i--; hilbert(d,r,u,l,i,h,x,y); move(r,h,x,y); hilbert(r,d,l,u,i,h,x,y); move(d,h,x,y); hilbert(r,d,l,u,i,h,x,y); move(l,h,x,y); hilbert(u,l,d,r,i,h,x,y); } } int main() { int n,x1,y1; int x0=50,y0=150,x,y,h=10,r=2,d=3,l=4,u=1; printf)"\nGive the value of n: "); scanf("%d",&n); x=x0;y=y0; int gm,gd=DETECT; initgraph(&gd,&gm,NULL); moveto(x,y); hilbert(r,d,l,u,n,h,x,y); ```	

<table>
<tr><td></td><td></td><td><pre>   delay(10000);
        closegraph();
        return 0;
}</pre></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td>Attempt any TWO of the following</td><td>12 M</td></tr>
<tr><td></td><td>a</td><td>Write a Program in ‘C’ for DDA Circle drawing algorithm</td><td>6 M</td></tr>
<tr><td></td><td>Ans</td><td><pre>#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
 int gdriver=DETECT,gmode,errorcode,tmp,i=1,rds;
 float st_x,st_y,x1,x2,y1,y2,ep;
 initgraph(&gdriver,&gmode,"C:\\TC\\BGI");
       printf("Enter Radius:");
 scanf("%d",&rds);
 while(rds>pow(2,i))
  i++;
   ep=1/pow(2,i);
 x1=rds; y1=0;
 st_x=rds; st_y=0;
 do
 { x2=x1+(y1*ep);
  y2=y1-(x2*ep);
  putpixel(x2+200,y2+200,10);
  x1=x2;
  y1=y2;
 }while((y1-st_y)<ep || (st_x-x1)>ep);
    getch();
}</pre></td><td>Correct Program 6 marks</td></tr>
<tr><td></td><td>b</td><td>Perform a 45° rotation of triangle A(0,0), B(1,1), C(5,2)
(i) About the origin (ii) About P(-1,-1)</td><td>6 M</td></tr>
<tr><td></td><td>Ans</td><td>About the Origin: -</td><td>Each Sub problem – 3 M</td></tr>
</table>

Solution: We can represent the given triangle, in matrix form, using homogeneous coordinates of the vertices:

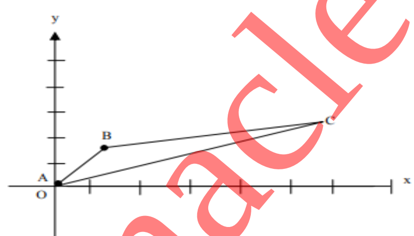
$$[ABC]=\begin{bmatrix} A & 0 & 0 & 1 \\ B & 1 & 1 & 1 \\ C & 5 & 2 & 1 \end{bmatrix}$$

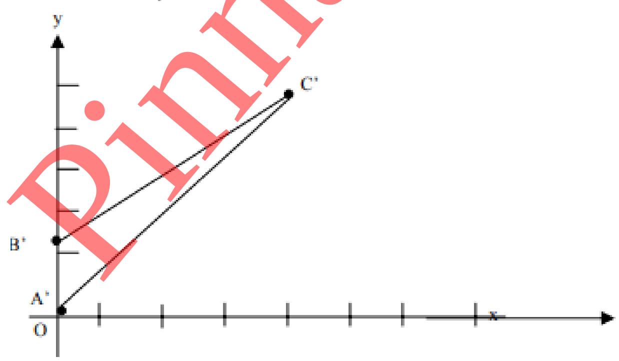
The matrix of rotation is: $R_\theta = R_{45}{}^{0} = \begin{bmatrix} \cos45^0 & \sin45^0 & 0 \\ -\sin45^0 & \cos45^0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

So the new coordinates A'B'C' of the rotated triangle ABC can be found as:

$$[A'B'C']=[ABC].\,R_{45^\circ}=\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{bmatrix}\begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & \sqrt{2} & 1 \\ 3\sqrt{2}/2 & 7\sqrt{2}/2 & 1 \end{bmatrix}$$

Thus $A'=(0,0)$, $B'=(0,\sqrt{2})$, $C'=(3\sqrt{2}/2, 7\sqrt{2}/2)$





	c	**Apply the Liang-Barsky algorithm to the line with co-ordinate (30,60) & (60,25) against the window: (Xmin, Ymin) = (10.10) & (Xmax,Ymax) = (50,50)**	**6 M**

<table>
<tr>
<td></td>
<td>Ans</td>
<td>
Given:

$(X_{min},Y_{min})=(10,10)$ and $(X_{max},Y_{max})=(50,50)$
P1 (30, 60) and P2 = (60, 25)

Solution:

Set Umin = 0 and Umax = 1

ULeft= q1 / p1
= X1 – Xmin / - Δ X

= 30 – 10 / - (60 - 30)

= 20 / - 30

= -0.67

URight= q2 / p2
= Xmax – X1/ ΔX

= 50 – 30 / (60 - 30)

= 20 / 30

= 0.67

UBottom= q3 / p3
= Y1 – Ymin / - ΔY

= 60 – 10 / - (25 – 60)

= 50 / 35

= 1.43

UTop= q4 / p4
= Ymax – Y1 / ΔY

= 50 – 60 / (25 - 60)

= -10 / - 35

= 0.29

Since ULeft=−0.57which is less than Umin. Therefore we ignore it.
Similarly UBottom=1.43 which is greater than Umax. So we ignore it.
URight=Umin = 0.67 (Entering)
UTop=Umax = 0.29 (Exiting)
We have UTop= 0.29 and URight= 0.67
Q – P = (ΔX, ΔY) = (30, -35)

Since Umin>Umax, there is no line segment to draw.
</td>
<td>Remark:
Calculation of each side 1 M;
Decision of displaying line coordinates with justification 2 M</td>
</tr>
</table>