Specify animation seq. using kinematic & inverse kinematic.

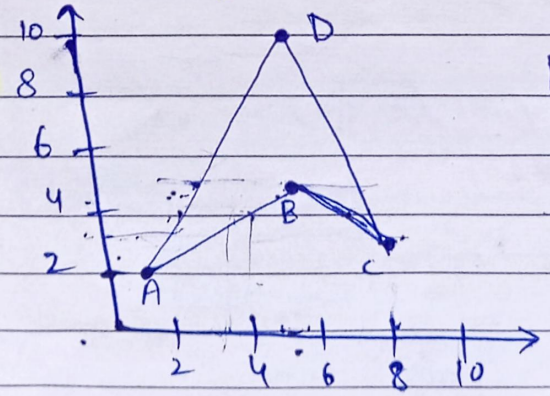
Kinematic : We specify the animation by giving motion parameters ( position, velocity & acc.) without reference to the forces that cause the motion. For constant velocity, (zero acc.), we designate the motions of rigid bodies in a scene by giving initial position & velocity vector for each object.

Inverse kinematics : We specify the initial & final positions of objects at specified times and the motion parameters are computed by the system.

Why z-buffer is a fast algo ?

1) No pre sorting is necessary
2) No object-object comparisons are required.

A(1, 2), B(5, 5) C(8, 3), D(5, 10). Trace the content of AET.




Increasing order of y :
A < C < B < D

creating horizontal scan lines

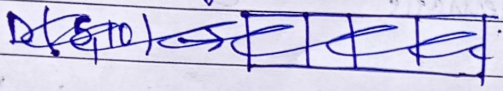
| $y_{max}$ | $x_{ymin}$ | $1/m$ |
|---|---|---|

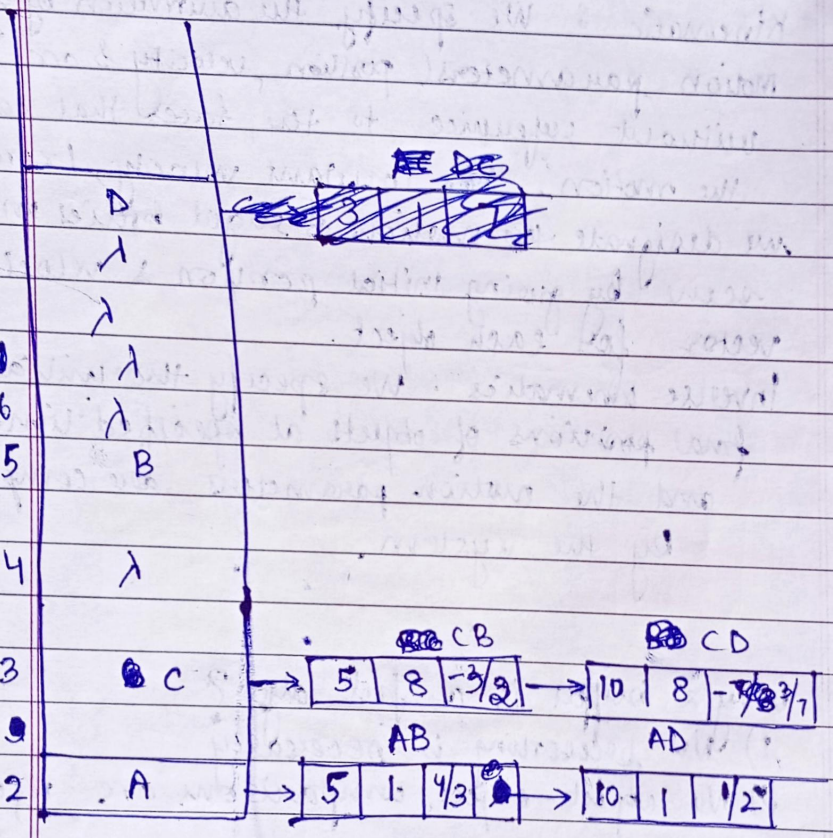
A(1,2) C(8,3) B(5,5) D(5,10)

PAGE NO.

DATE : / /

Global edge table :






5-1/5-2 4/3

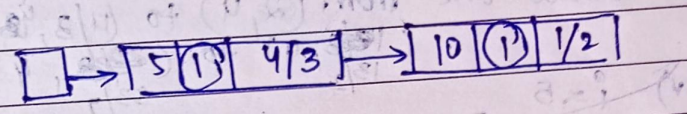
5-1/10-2 4/8

3-5/8-5 3/3

PAGE NO.
DATE : / /

AET :

i) i = 2



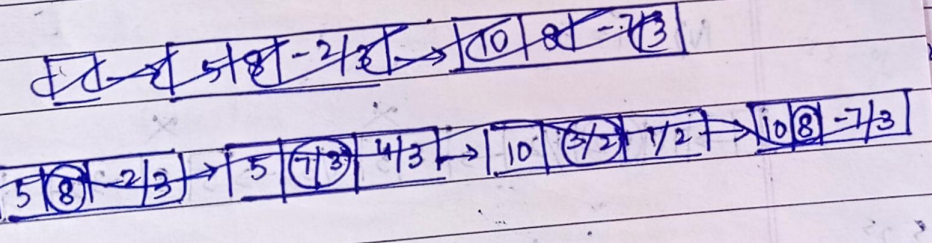

draw blue line {(1, i), (1, i)}

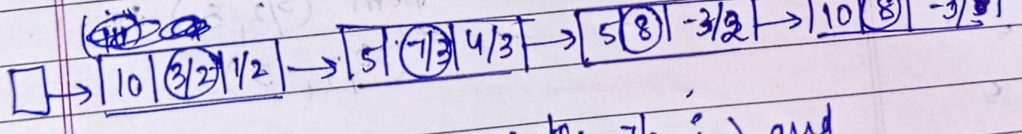
i.e. (1, 2) and (1, 2) are to be drawn

$x = x + 1/m$

ii) i = 3

keep in order of x




$1 + \frac{4}{3}$

$\frac{3+4}{3}$

draw one {(3/2, i) to (7/3, i) and
(8, i) to (8, i)}

i.e. (3/2, 3) to (7/3, 3) and
(8, 3) to (8, 3) to be drawn

$\frac{2+1}{2}$ 2

$8 - \frac{2}{3}$

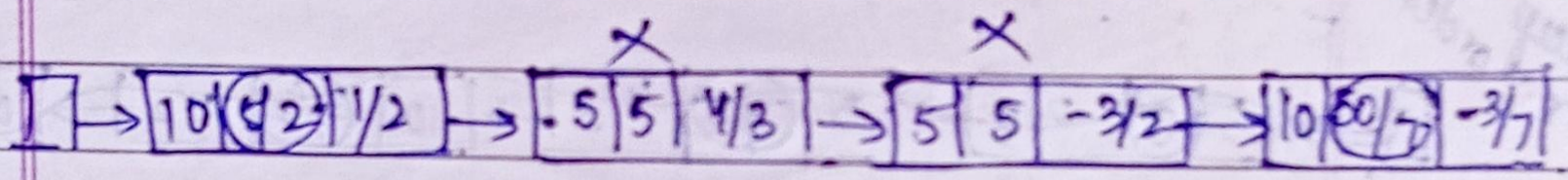
...m requires that we have
...e buffer F in which col
...es a z-buffer Z, with the
...which a z-value is stored
...n initialized to zero,
...e at the back of the
...frame buffer is initialized
Polygons are scan converted
in an arbitrary order.
...ion process, if the polyg
...d at (x, y) is no farther
...is the point whose col
...y in the buffer, then
depth replace the ol
...me buffer. record the
...the largest z
...each (x, y). Thus the
...een in the order in
...d.

...++) {
...x++) {
BACKGROUND_VALUE)

/* clear fram
z-buffer

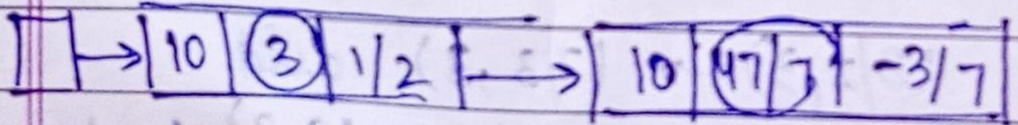
/* Draw polygons */

polygon's projection)



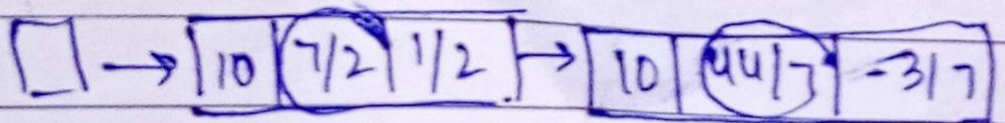

∴ Draw line from $(5/2, 5)$ to $(50/7, 5)$

(v) $i = 6$




∴ Draw $(3, 6)$ to $(47/7, 6)$

(vi) $i = 8$



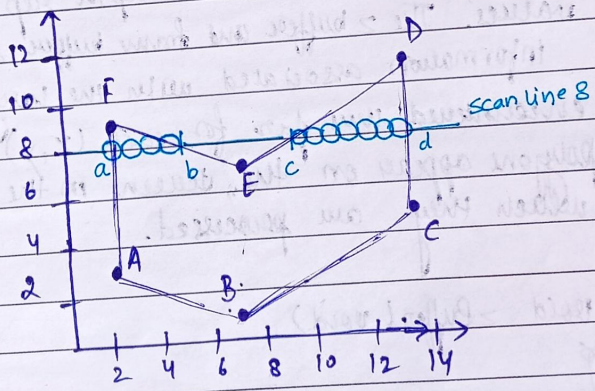

$(7/2, 8)$ to $(44/7, 8)$

Q8. a) The z-buffer algorithm requires that we have available not only a frame buffer F in which color values are stored, but also a z-buffer Z; with the same no. of entries in which a z-value is stored for each pixel. The z-buffer is initialized to zero, representing the z-value at the back of the clipping plane, and the frame buffer is initialized to the background color. Polygons are scan converted into the frame buffer in an arbitrary order. During the scan conversion process, if the polygon point being scan converted at $(x, y)$ is no farther from the viewer than is the point whose color and depth are currently in the buffers, then the new point's color and depth replace the old values. The z buffer and frame buffer record the information associated with the largest z encountered thus far for each $(x, y)$. Thus the polygons appear on the screen in the order in which they are processed.

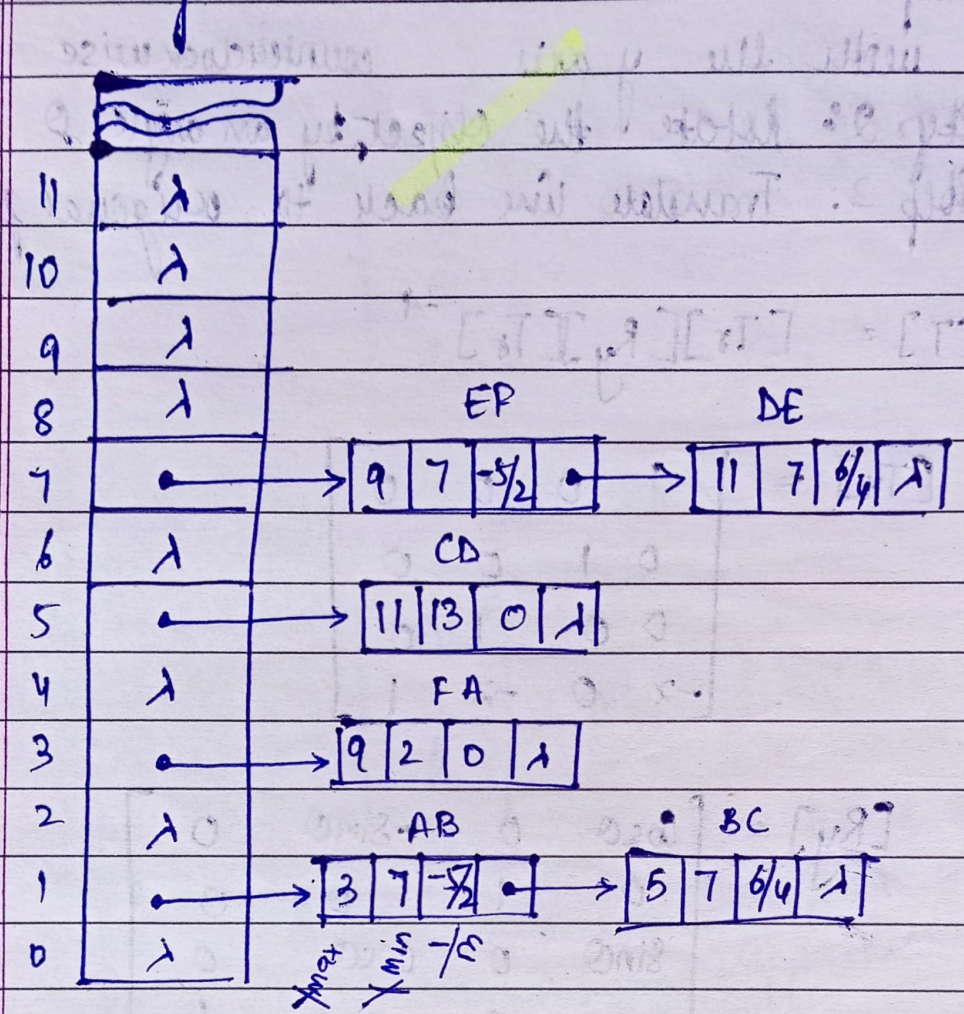
```
void zBuffer(void)
{
    int x, y;
    for(y=0; y < XMAX; y++) {        /* clear frame and z-buffer */
        for(x=0; x < XMAX; x++) {
            WritePixel(x, y, BACKGROUND_VALUE);
            WriteZ(x, y, 0)
        }
    }
    for(each polygon) {              /* Draw polygons */
        for(each pixel in polygon's projection) {
```

```
double pz = polygon's z value at pixel
            coords (x, y);
if (pz >= Read Z (x, y)) {   /* New point is
                                not farther */
    Write z (x, y, pz);
    WritePixel (x, y, polygon's color at
                pixel coords (x, y));
}
}
}
}   /* zBuffer */
```

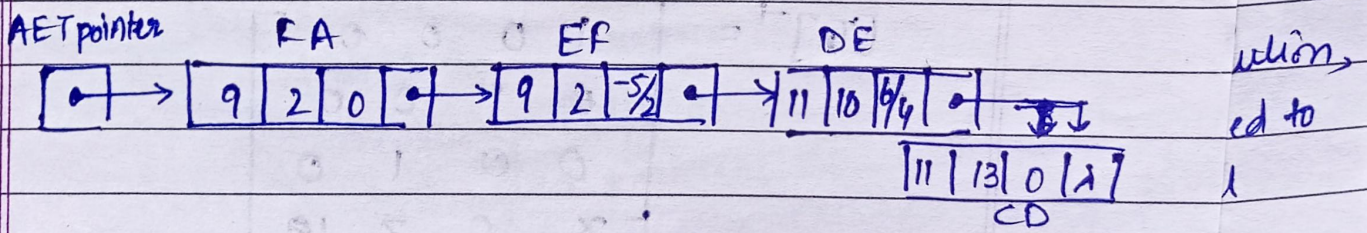
Q3. (a) Polygon :

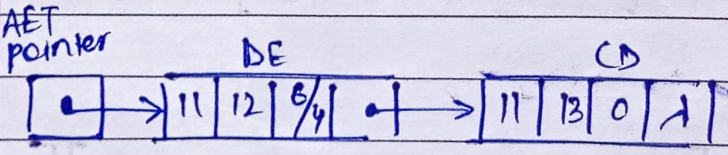



1) Edge table :




2) AET table for scan line 9 and scan line 10 :



$$[T_{TL}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x & 0 & -z & 1 \end{bmatrix}$$

$$[R_y] = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[T_r]^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & 0 & z & 1 \end{bmatrix}$$

$$\therefore [T] = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ -x\cos\theta & 0 & x\sin\theta & 1 \\ -z\sin\theta & & -z\cos\theta & \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & 0 & z & 1 \end{bmatrix}$$

Q11 (b) Curve fitting techniques are used to specify the animation paths b/w key frames.

constant + positive + deacceleration.
(zero)

(ii) Geometric vector $G_B$:

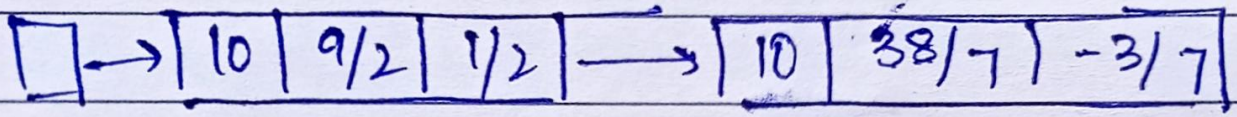
$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

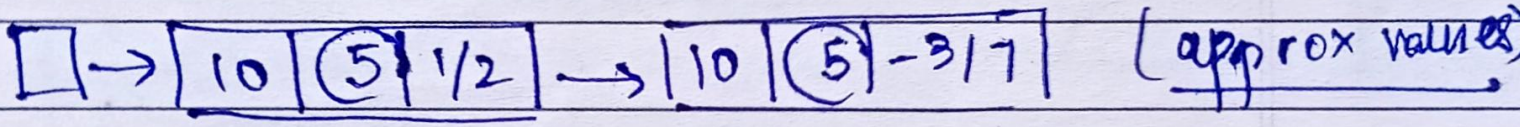
where $P_1, P_2, P_3, P_4$ are the four points, two end control points ~~and two~~ and two points which are approximated.

05. (b) (ii) Dithering

Primarily, it refers to techniques for approximating halftones without reducing resolution, as pixel grid patterns do. It is also applied to halftone approximation methods using pixel grids and sometimes used to refer to color halftone.

(imp) $y = 10$ ($y_{min} = y_{max}$)




(approx values)

∴ Draw Point (5, 10)

(ix)

Deleting all nodes.