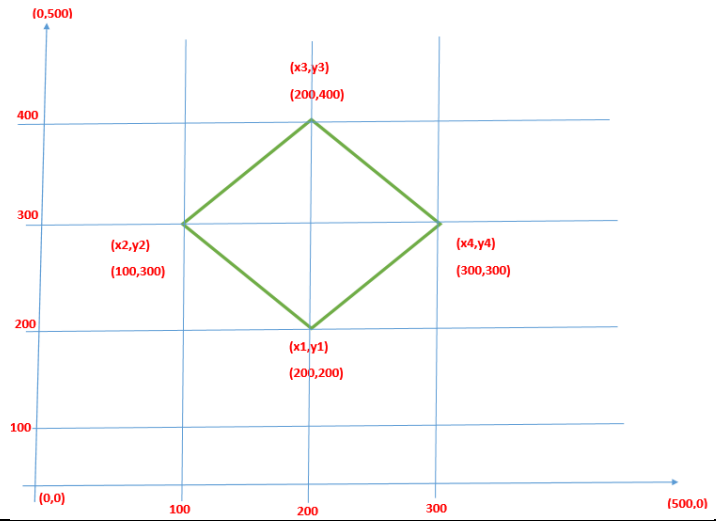
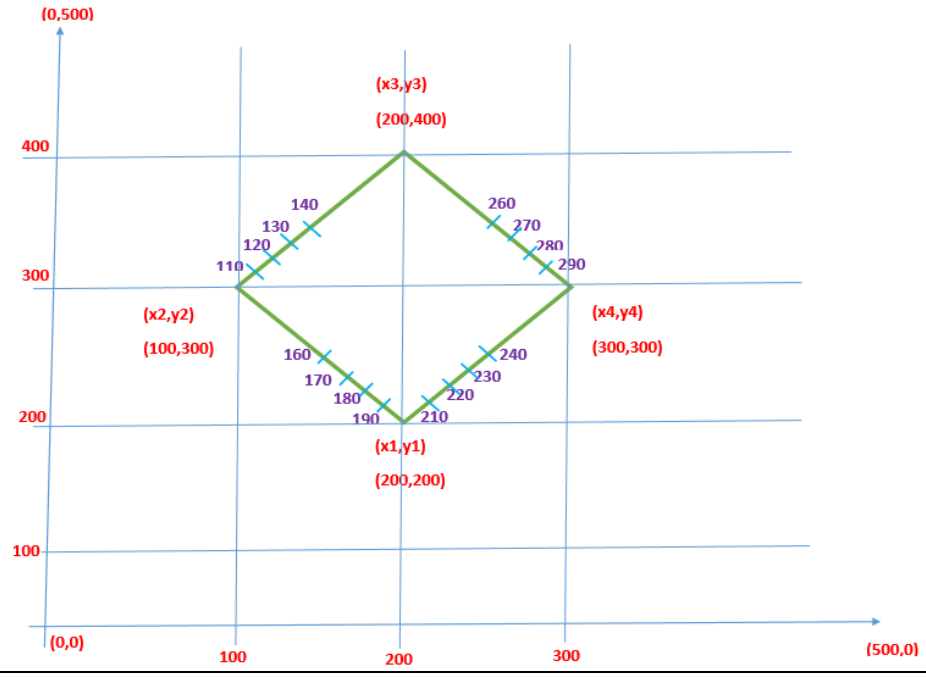
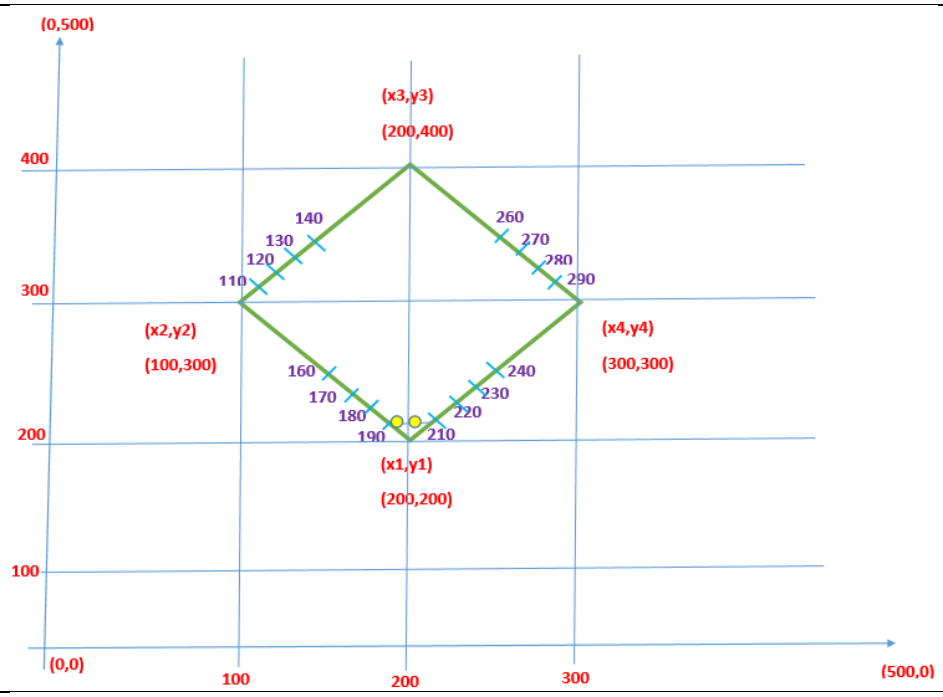
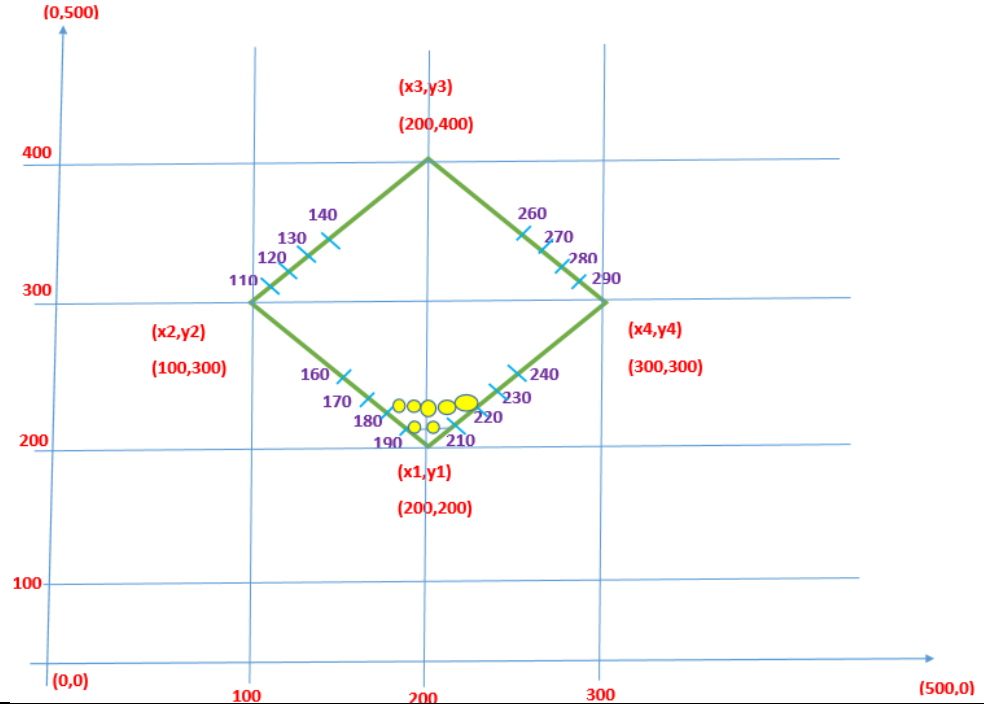
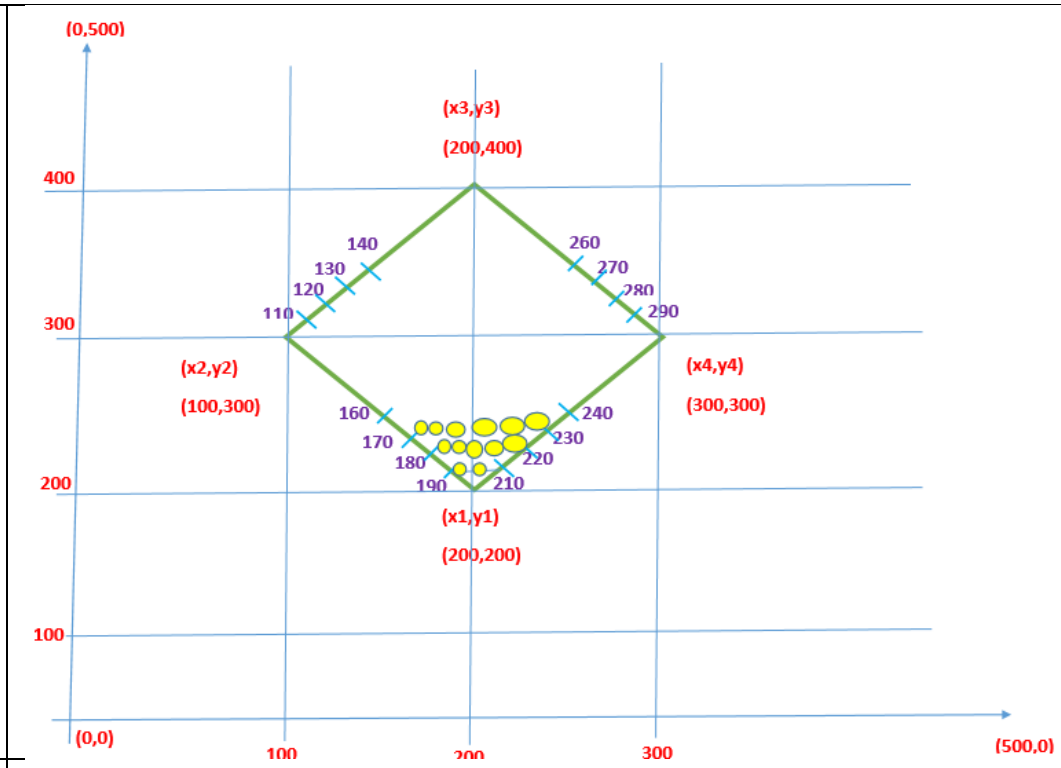
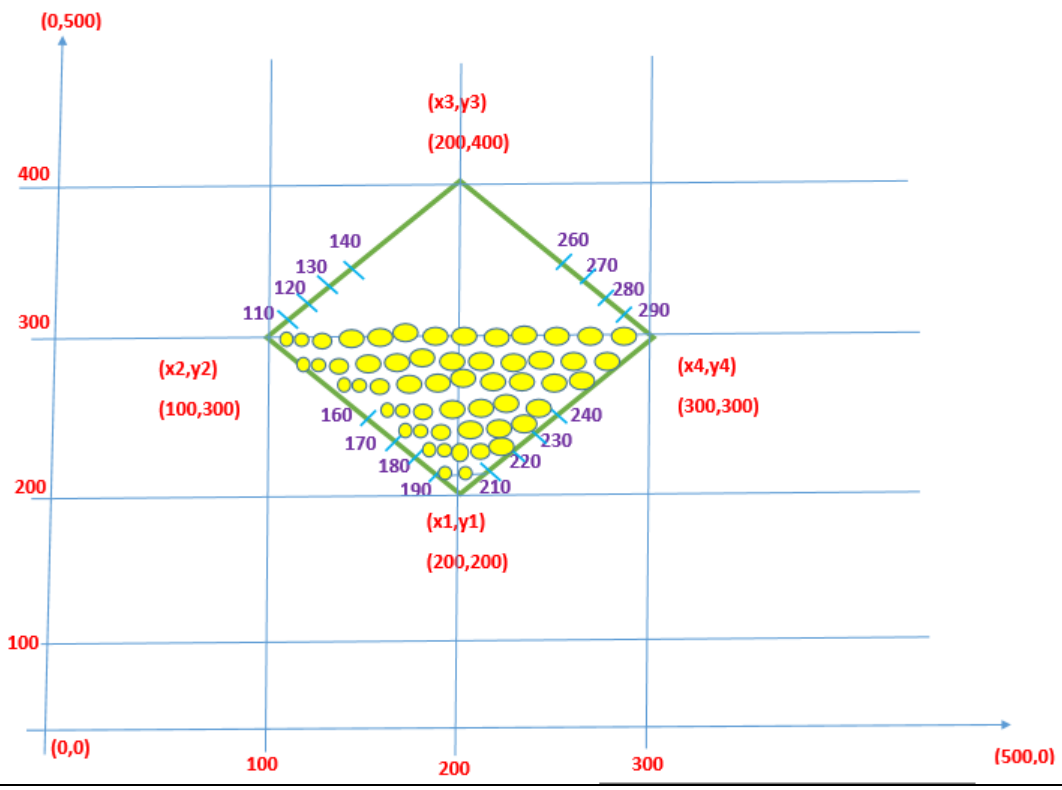
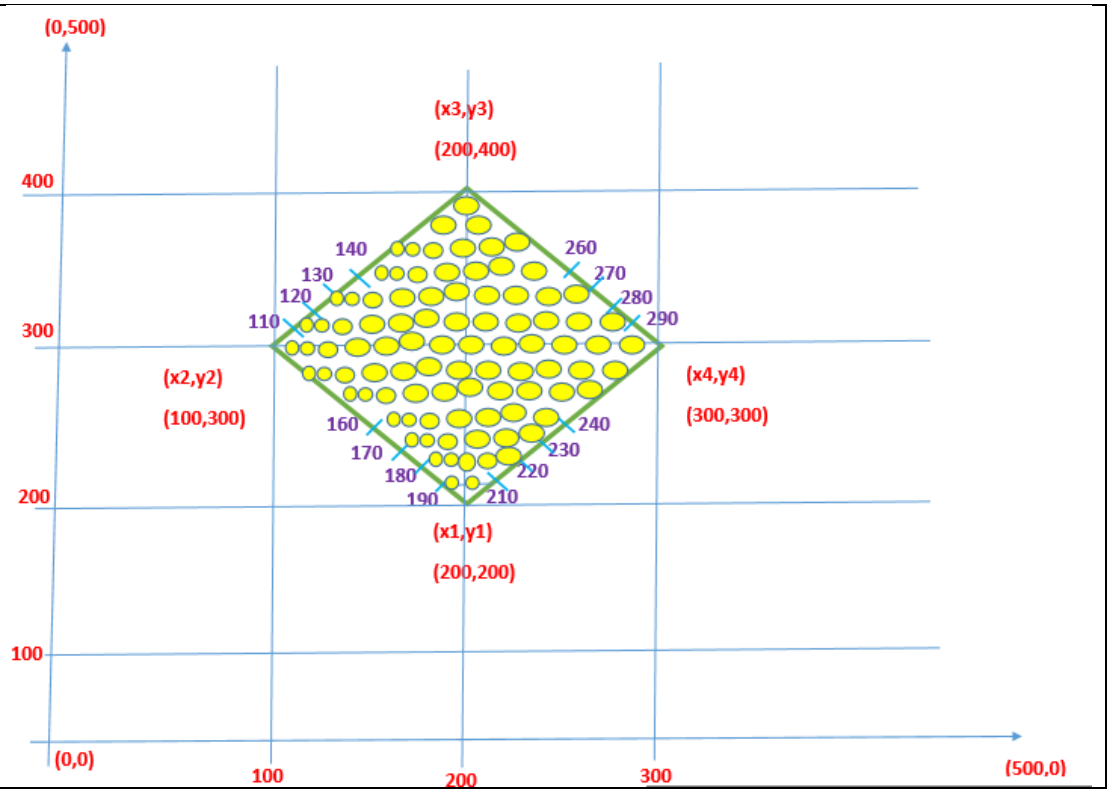
9. Develop a menu driven program to fill any given polygon using scan-line area filling algorithm.

Step	Description	Output
Step 1	Define the boundary -> x axis and y axis Also, define your polygon i,e., x1,y1 x2,y2 x3,y3 x4,y4	
Step 2	Run 4 big for loops and mark your left edges and right edges	

Step 3	Start filling from the left edge to right edge	
		


(0,500)
(x3,y3)
(200,400)
400
140
130
120
110
260
270
280
290
300
(x2,y2)
(100,300)
(x4,y4)
(300,300)
160
170
180
190
240
230
220
210
200
(x1,y1)
(200,200)
100
(0,0)
100
200
300
(500,0)


(0,500)
(x3,y3)
(200,400)
400
140
130
120
110
260
270
280
290
300
(x2,y2)
(100,300)
(x4,y4)
(300,300)
160
170
180
190
240
230
220
210
200
(x1,y1)
(200,200)
100
(0,0)
100
200
300
(500,0)

Step 4	STOP once you fill all the rows of left edge to right edge	

You remember I had taught all the iterations via debug - breakpoints and an excel sheet which kinda looked like this after 4th iteration?

i	leftedge	rightedge
198	500	0
199	500	0
200	200	200
201	199	201
202	198	202
203	197	203
204	196	204
205	195	205
206	194	206
207	193	207
208	192	208
209	191	209
210	190	210
211	189	211
212	188	212
213	187	213
214	186	214
215	185	215

and so on….

```
#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>

float x1, x2, x3, x4, y1, y2, y3, y4;   // our polygon has 4 lines - so 8 coordinates

void edgedetect(float x1, float y1, float x2, float y2, int *left_edge, int *right_edge)
{
    float x_slope, x, temp;
    int i;

    if ((y2-y1)<0)   // decide where to start
    {
        temp = y1;
        y1 = y2;
        y2 = temp;

        temp = x1;
        x1 = x2;
        x2 = temp;
    }

    if ((y2-y1)!=0)                          // compute the values
        x_slope = (x2 - x1) / (y2 - y1);
    else
        x_slope = x2 - x1;

    x = x1;

    for (i = y1; i <= y2; i++)  // fill the values
    {
        if (x < left_edge[i])
            left_edge[i] = x;

        if (x > right_edge[i])
            right_edge[i] = x;

        x = x + x_slope;
    }
}

void draw_pixel (int x, int y)      // fill the polygon point by point (pixel by pixel)
{
    glColor3f (1, 1, 0);            // fill the RHOMBUS in yellow colour
    glBegin (GL_POINTS);
         glVertex2i (x, y);
    glEnd ();
}
```

```
void scanfill (float x1, float y1, float x2, float y2, float x3, float y3, float x4, float y4)
{
    int left_edge[500], right_edge[500];
    int i, y;

    for (i = 0; i <= 500; i++)
    {
        left_edge [i] = 500;     // fill all the left_edge values as 500 initially
        right_edge [i] = 0;      // fill all the right_edge values as 0 initially
    }

    edgedetect (x1, y1, x2, y2, left_edge, right_edge);  // first line

    edgedetect (x2, y2, x3, y3, left_edge, right_edge);  // second line

    edgedetect (x3, y3, x4, y4, left_edge, right_edge);  // third line

    edgedetect (x4, y4, x1, y1, left_edge, right_edge);  // fourth line

    for (y = 0; y <= 500; y++)     // now that you have calculated all values, start filling
    {                                  from left edge to right edge row by row pixel by pixel
        if (left_edge[y] <= right_edge[y])
        {
            for (i = left_edge[y]; i <= right_edge[y]; i++)
            {
                draw_pixel (i, y);
                glFlush ();
            }
        }
    }
}

void display()
{
    x1 = 200, y1 = 200;                  // RHOMBUS coordinates
    x2 = 100, y2 = 300;
    x3 = 200, y3 = 400;
    x4 = 300, y4 = 300;

    glClear (GL_COLOR_BUFFER_BIT);

    glColor3f (0, 0, 1);                 // blue RHOMBUS

    glBegin (GL_LINE_LOOP);              // draw the RHOMBUS
        glVertex2f (x1, y1);
        glVertex2f (x2, y2);
        glVertex2f (x3, y3);
        glVertex2f (x4, y4);
    glEnd ();
```

```
    scanfill (x1, y1, x2, y2, x3, y3, x4, y4);      // FILL the RHOMBUS
}

void init()
{
    glClearColor (1, 1, 1, 1);
    gluOrtho2D (0, 499, 0, 499);
}

int main (int argc, char** argv)
{
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (0, 0);
    glutCreateWindow ("Filling a Polygon using Scan-line Algorithm");

    init ();

    glutDisplayFunc (display);

    glutMainLoop ();
}
```

****

OUTPUT