EXPERIMENT 5:

SCALING:

```
#include <GL/glut.h>
#include <iostream>
#include <math.h>

using namespace std;

void findNewCoordinate(int s[][2], int p[][1])
{
   int temp[2][1] = { 0 };

   for (int i = 0; i < 2; i++)
      for (int j = 0; j < 1; j++)
         for (int k = 0; k < 2; k++)
            temp[i][j] += (s[i][k] * p[k][j]);

   p[0][0] = temp[0][0];
   p[1][0] = temp[1][0];
}

void scale(int x[], int y[], int sx, int sy)
{
   // Triangle before Scaling
   glBegin(GL_LINE_LOOP);
   glVertex2f(x[0], y[0]);
   glVertex2f(x[1], y[1]);
   glVertex2f(x[1], y[1]);
   glVertex2f(x[2], y[2]);
   glVertex2f(x[2], y[2]);
```

```
    glVertex2f(x[0], y[0]);
    glEnd();

    int s[2][2] = { sx, 0, 0, sy };
    int p[2][1];

    // Scaling the triangle
    for (int i = 0; i < 3; i++)
    {
        p[0][0] = x[i];
        p[1][0] = y[i];

        findNewCoordinate(s, p);

        x[i] = p[0][0];
        y[i] = p[1][0];
    }
    glBegin(GL_LINE_LOOP);
    glVertex2f(x[0], y[0]);
    glVertex2f(x[1], y[1]);
    glVertex2f(x[1], y[1]);
    glVertex2f(x[2], y[2]);
    glVertex2f(x[2], y[2]);
    glVertex2f(x[0], y[0]);
    glEnd();
    glFlush();
    //cout<< x[0] <<" " << x[1] << " " << x[2];
    //cout << "..........";
    //cout << y[0] << " " << y[1] << " " << y[2];

}
```

```
void init(void)
{
   glClearColor(0.0, 0.0, 0.0, 0.0);
   gluOrtho2D(0, 500, 0, 500);
}

int main(int argc, char** argv)
{
   int x[] = { 100, 200, 300 };
   int y[] = { 200, 100, 200 };
   int sx = 2, sy = 2;

   int gd, gm;
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize(640, 480);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("2D Transformaton Scaling ");
   init();

   scale(x, y, sx, sy);

   glutMainLoop();
   return 0;
}
```

TRANSLATION:

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

```
#include <iostream>

#include <math.h>


using namespace std;



void translateRectangle(int P[][2], int T[])

{


   glBegin(GL_LINE_LOOP);

   glVertex2f(P[0][0], P[0][1]);

   glVertex2f(P[1][0], P[0][1]);

   glVertex2f(P[1][0], P[1][1]);

   glVertex2f(P[0][0], P[1][1]);

      glEnd();

   // calculating translated coordinates

   P[0][0] = P[0][0] + T[0];

   P[0][1] = P[0][1] + T[1];

   P[1][0] = P[1][0] + T[0];

   P[1][1] = P[1][1] + T[1];



   glBegin(GL_LINE_LOOP);

   glVertex2f(P[0][0], P[0][1]);

   glVertex2f(P[1][0], P[0][1]);

   glVertex2f(P[1][0], P[1][1]);

   glVertex2f(P[0][0], P[1][1]);

   glEnd();

   glFlush();
}


void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    gluOrtho2D(0, 500, 0, 500);
}

int main(int argc, char** argv)
{
    int P[2][2] = { 50, 80, 120, 180 };
    int T[] = { 100, 100 }; // translation factor

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("2D Transformaton Scaling ");
    init();

    translateRectangle(P, T);

    glutMainLoop();
    return 0;
}
```