```
#include<windows.h>
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;

#define w 500
#define h 500

void init()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glClearColor(1.0, 1.0, 1.0, 0.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(-w / 2, w / 2, -h / 2, h / 2);
}

void setpixel(GLint x, GLint y)
{
        glColor3f(1.0, 1.0, 1.0);
        glPointSize(4.0);
        glBegin(GL_POINTS);
        glVertex2f(x, y);
        glEnd();
        glFlush();
}

float mati[30][2], mato[30][2];
int k = 0, c;

class trans
{
        int transl[2][2];
public:
        int mul(float s[2][2])
        {
                int i, j, k;
                float sum;
                for (i = 0; i < c; i++)
                {
                        for (j = 0; j < 2; j++)
                        {
                                sum = 0;
                                for (k = 0; k < 2; k++)
                                        sum = sum + mati[i][k] * s[k][j];
                                mato[i][j] = sum;
                        }
                }
                return mato[30][2];
        }
```

```
        void plot(float m[][2], float r, float g, float b)
        {
                glColor3f(r, g, b);
                glBegin(GL_LINE_LOOP);
                for (int i = 0; i < c; i++)
                        glVertex2f(m[i][0], m[i][1]);
                glEnd();
                glFlush();
        }
        void translation(float tx, float ty)
        {
                int i;
                for (i = 0; i < c; i++)
                {
                        mato[i][0] = mati[i][0] + tx;
                        mato[i][1] = mati[i][1] + ty;
                }
        }
        void scaling()
        {
                float scale[2][2];
                int i;
                float sx, sy;
                cout << "\nScaling:\nEnter Sx factor::";
                cin >> sx;
                cout << "\nEnter Sy factor::";
                cin >> sy;
                scale[0][0] = sx;
                scale[0][1] = 0;
                scale[1][0] = 0;
                scale[1][1] = sy;
                mul(scale);
                plot(mato, 0.0, 1.0, 0.0);
        }
        void rotation()
        {
                int rot;
                float angle, rota[2][2];
                cout << "\nRotation:\nEnter angle::";
                cin >> angle;
                angle = (3.14 * angle) / 180;
                cout << "1.For Anti-Clockwise rotation\n2.For Clockwise
rotation\nEnter your choice::";
                cin >> rot;
                switch (rot)
                {
                case 1:rota[0][0] = cos(angle);
                        rota[0][1] = -sin(angle);
                        rota[1][0] = sin(angle);
                        rota[1][1] = cos(angle);
```

```
                        break;
                case 2:rota[0][0] = cos(angle);
                        rota[0][1] = sin(angle);
                        rota[1][0] = -sin(angle);
                        rota[1][1] = cos(angle);
                        break;
                default:cout << "\nInvalid Input!!!";
                        system("pause");
                        exit(0);
                }
                mul(rota);
        }
        void reflection()
        {
                char axis;
                int i;
                cout << "\nEnter reflection axis:";
                cin >> axis;
                glBegin(GL_LINE_LOOP);
                if (axis == 'x' || axis == 'X')
                {
                        for (i = 0; i < c; i++)
                        {
                                glVertex2i(round(mati[i][0]), round(mati[i][1] *
-1));
                        }
                }
                else if (axis == 'y' || axis == 'Y')
                {
                        for (i = 0; i < c; i++)
                        {
                                glVertex2i(round(mati[i][0] * -1),
round(mati[i][1]));
                        }
                }
                glEnd();
        }

        void shearing()
        {
                char axis;
                int i, shearingX, shearingY;
                cout << "\nEnter shearing axis:";
                cin >> axis;
                glColor3f(0.0, 1.0, 0.0);
                glBegin(GL_LINE_LOOP);
                if (axis == 'x' || axis == 'x')
                {
                        cout << "\nEnter shearing factor for x:";
                        cin >> shearingX;
```

```
                        glVertex2i(mati[0][0], mati[0][1]);
                        glVertex2i(mati[1][0] + shearingX, mati[1][1]);
                        glVertex2i(mati[2][0] + shearingX, mati[2][1]);
                        glVertex2i(mati[3][0], mati[3][1]);
                }
                else if (axis == 'y' || axis == 'y')
                {
                        cout << "\nEnter shearing factor for y:";
                        cin >> shearingY;
                        glVertex2i(mati[0][0], mati[0][1]);
                        glVertex2i(mati[1][0], mati[1][1]);
                        glVertex2i(mati[2][0], mati[2][1] + shearingY);
                        glVertex2i(mati[3][0], mati[3][1] + shearingY);
                }
                //}
                glEnd();
        }
}t;

void menu(int c)
{
        float tx, ty;
        if (c == 1)
        {
                cout << "\nTranslation->\nEnter tx factor::";
                cin >> tx;
                cout << "\nEnter ty factor::";
                cin >> ty;
                t.translation(tx, ty);
                t.plot(mato, 1.0, 0.0, 0.0);
                cout << "Translation object in red color \n";
        }
        else if (c == 2)
        {
                t.scaling();
                cout << "Scaled object in green color\n";
        }
        else if (c == 3)
        {
                t.rotation();
                t.plot(mato, 0.0, 1.0, 1.0);
                cout << "Rotated object in light blue\n";
        }
        else if (c == 4)
        {
                t.reflection();
                t.plot(mati, 1.0, 1.0, 0.0);
                cout << "Reflection Done\n";
        }
        else if (c == 5)
```

```
        {
                t.shearing();
                t.plot(mato, 1.0, 0.0, 1.0);
                cout << "Shearing Done\n";
        }
        else if (c == 6)
        {
                glClearColor(0, 0, 0, 1);
                glClear(GL_COLOR_BUFFER_BIT);
                glFlush();
        }
        else if (c == 7)
        {
                exit(1);
        }
        else
        {
                cout << "\nInvalid option. Try again.";
        }
}

void keyboard(unsigned char key, int x, int y)
{
        if (key == 27)
                exit(0);

        if (key == 108 || key == 76)
        {
                glColor3f(0.0, 0.0, 1.0);
                glBegin(GL_LINE_LOOP);
                for (int i = 0; i < k; i++)
                        glVertex2f(mati[i][0], mati[i][1]);
                glEnd();
                c = k;
                k = 0;
                glFlush();
        }
}

void mouse(int button, int state, int x, int y)
{
        int x1, y1, p;
        if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
                mati[k][0] = (float)(x - 250);
                mati[k][1] = (float)(250 - y);
                glColor3f(1.0, 0.0, 0.0);
                glPointSize(3.0);
                glBegin(GL_POINTS);
                glVertex2f(mati[k][0], mati[k][1]);
                glEnd();
```

```
                k++;
                glFlush();
        }
        glutCreateMenu(menu);
        glutAddMenuEntry("TRANSLATION", 1);
        glutAddMenuEntry("SCALING", 2);
        glutAddMenuEntry("ROTATION", 3);
        glutAddMenuEntry("REFLECTION", 4);
        glutAddMenuEntry("SHEARING", 5);
        glutAddMenuEntry("CLEAR", 6);
        glutAddMenuEntry("EXIT", 7);
        glutAttachMenu(GLUT_RIGHT_BUTTON);
}

int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(w, h);
        glutCreateWindow("2D Transformations");
        cout << "Use mouse pointer to put point on the graphics screen\nUse 'l' or
'L' to make polygon on graphics window.\nUse right mouse button to get 2D
transformations menu.";
        cout<<"blue one is original and other one is transformed one"<<endl;
        glutDisplayFunc(init);
        glutKeyboardFunc(keyboard);
        glutMouseFunc(mouse);
        glutMainLoop();

        return 0;
}
```