/* **Experiment No. 3**

Problem Statement : Implement following 2D transformations on the object with respect to axis :

1:Ttranslation() 2: Scaling() 3:Shearing() 4: Rotation() 5: Reflection()

Roll No. : ITSA16

Name : KRUSHNA SUDHIR SHINDE

*/

Program Code :

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

using namespace std;

int A[20][3], C[20][3], tx, ty, sx, sy, shx, shy, n;
float B[3][3];

void lineDDA(int x1, int y1, int x2, int y2);
void translation();
void scaling();
void shearing();
void rotation();
void reflection();
void multiply();
void display();
void unit();

void Init() {
   glClearColor(1.0, 1.0, 1.0, 0.0);
   gluOrtho2D(0, 1280, 0, 1024);
}

void display() {
  int i;
  glPointSize(3);
  glLineWidth(3);

   glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,0.0,0.0);
lineDDA(320,0,320,480);
```

```
lineDDA(0,240,640,240);
glColor3f(0.0,0.0,1.0);
for(i=0;i<n-1;i++)
{
lineDDA(A[i][0]+320,A[i][1]+240,A[i+1][0]+320,A[i+1][1]+240);
}
lineDDA(A[n-1][0]+320,A[n-1][1]+240,A[0][0]+320,A[0][1]+240);
glColor3f(0.0,1.0,0.0);
for(i=0;i<n-1;i++)
{
lineDDA(C[i][0]+320,C[i][1]+240,C[i+1][0]+320,C[i+1][1]+240);
}
lineDDA(C[n-1][0]+320,C[n-1][1]+240,C[0][0]+320,C[0][1]+240);
   glFlush();
}

void unit() {
   for (int i = 0; i < 3; i++) {
      for (int j = 0; j < 3; j++) {
         B[i][j] = (i == j) ? 1 : 0;
      }
   }
}

void multiply() {
int i, j, k;
   for ( i = 0; i < n; i++) {
      for ( j = 0; j < 3; j++) {
         C[i][j] = 0;
         for ( k = 0; k < 3; k++) {
            C[i][j] += A[i][k] * B[k][j];
         }
      }
      C[i][2] = 1;
   }
  for (i =0; i<n; i++){
     for(j=0; j<3; j++){
       cout<<C[i][j]<< "  ";
     }
     cout<<endl;
   }

}

void lineDDA(int x1, int y1, int x2, int y2) {
   int i, dx, dy, steps;
   float incx, incy, x, y;
```

```
    dx = x2 - x1;
    dy = y2 - y1;

    steps = (abs(dx) > abs(dy)) ? abs(dx) : abs(dy);
    incx = (float)dx / steps;
    incy = (float)dy / steps;
    x = x1;
    y = y1;

    glBegin(GL_POINTS);
    glVertex2f(x, y);
    for (i = 1; i <= steps; i++) {
        x += incx;
        y += incy;
        glVertex2f(round(x), round(y));
    }
    glEnd();
}


void translation() {
    cout << "\nEnter the value of tx and ty"<<endl;
    cin >> tx >> ty;
    unit();
    B[2][0] = tx;
    B[2][1] = ty;
    multiply();
}

void scaling() {
    cout << "\nEnter the value of sx and sy"<<endl;
    cin >> sx >> sy;
    unit();
    B[0][0] = sx;
    B[1][1] = sy;
    multiply();
}

void shearing() {
    int ch;

    cout << "\n Menu\n1. X axis\n2. Y axis\n3. XY line"<<endl;
    cout << "\nEnter the choice :";
    cin >> ch;
    unit();

    if (ch == 1) {
```

```
        cout << "\nEnter the value of shx"<<endl;
        cin >> shx;
        B[1][0] = shx;
    } else if (ch == 2) {
        cout << "\nEnter the value of shy"<<endl;
        cin >> shy;
        B[0][1] = shy;
    } else if (ch == 3) {
        cout << "\nEnter the value of shx\n";
        cin >> shx;
        B[1][0] = shx;
        cout << "\nEnter the value of shy\n";
        cin >> shy;
        B[0][1] = shy;
    } else {
        cout << "\nWrong choice\n";
        return;
    }
    multiply();
}

void rotation() {
    float t, ch1;
    cout << "\nEnter the value of angle:";
    cin >> t;
    float r = (t * 3.14) / 180;
    unit();

    cout << "\nMenu\n1. Clockwise Rotation\n2. Anticlockwise Rotation\nEnter your choice:
";
    cin >> ch1;

    B[0][0] = B[1][1] = cos(r);
    B[1][0] = (ch1 == 1) ? sin(r) : -sin(r);
    B[0][1] = (ch1 == 1) ? -sin(r) : sin(r);
    multiply();
}

void reflection() {
    int ch;
    cout << "\nMenu\n1. About x axis\n2. About y axis\n3. About X=Y line\n4. About x=-y
line\n5. About Origin\nEnter choice :";
    cin >> ch;
    unit();

    switch (ch) {
        case 1:
```

```
        B[1][1] = -1;
        break;

        case 2:
        B[0][0] = -1;
        break;

        case 3:
        B[0][0] = B[1][1] = 0;
        B[0][1] = B[1][0] = 1;
        break;

        case 4:
        B[0][0] = B[1][1] = 0;
        B[1][0] = B[0][1] = -1;
        break;

        case 5:
        B[0][0] = B[1][1] = -1;
        break;

        default: cout << "\nWrong choice\n"; return;
    }
    multiply();
}
int main(int argc, char **argv) {
    int i, ch;
    cout << "\nTransformation\n";
    cout << "\n1. Translation\n2. Scaling\n3. Shearing\n4. Rotation\n5. Reflection\nEnter your
choice: ";
    cin >> ch;

    cout << "\nEnter the number of vertices: ";
    cin >> n;
    for (i = 0; i < n; i++) {
        cout << "\nEnter the value of  x"<< i + 1 << " and  y" << i + 1 << ": "<<endl;
        cin >> A[i][0] >> A[i][1];
        A[i][2] = 1;
    }

    switch (ch) {
        case 1: translation(); break;
        case 2: scaling(); break;
        case 3: shearing(); break;
        case 4: rotation(); break;
        case 5: reflection(); break;
        default: cout << "\nWrong choice\n"; return 0;
```

```
    }

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("2D - Transformation");
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(1280, 1024);
    Init();
    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```