Practical -6

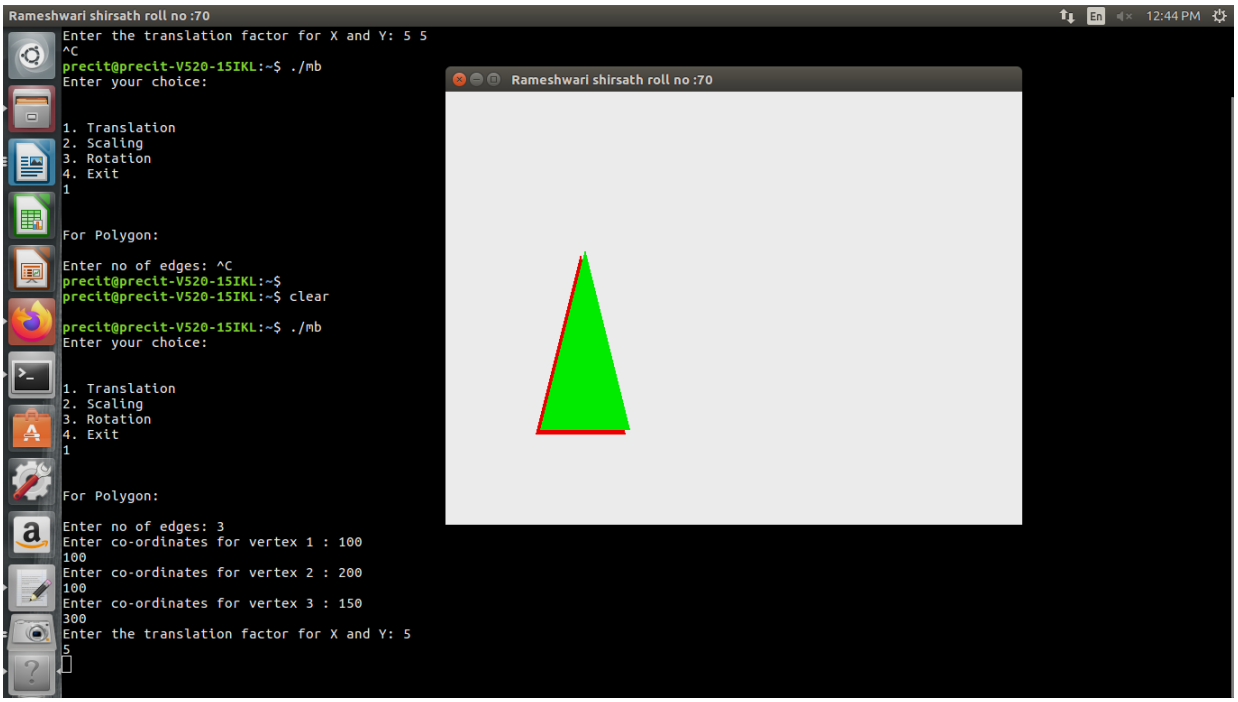
```
#include<math.h>
#include<iostream>
#include<vector>
#include<GL/glut.h>
using namespace std;
int pntX1, pntY1, choice = 0, edges;
vector<int> pntX;
vector<int> pntY;
int transX, transY;
double scaleX, scaleY;
int angleRad;
double round(double d)
{
return floor(d + 0.5);
}
void drawPolygon()
{
glBegin(GL_POLYGON);
glColor3f(1.0, 0.0, 0.0);
for(int i = 0; i < edges; i++)
{
glVertex2i(pntX[i], pntY[i]);
}
glEnd();
}
void drawPolygonTrans(int x, int y)
{
glBegin(GL_POLYGON);
glColor3f(0.0, 1.0, 0.0);
for(int i = 0; i < edges; i++)
{
glVertex2i(pntX[i] + x, pntY[i] + y);
}
glEnd();
}
void drawPolygonScale(double x, double y)
{
glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 1.0);
for(int i = 0; i < edges; i++)
{
glVertex2i(round(pntX[i] * x), round(pntY[i] * y));
}
glEnd();
}void myInit (void)
{
glClearColor(1.0, 1.0, 1.0, 0.0);
glColor3f(0.0f, 0.0f, 0.0f);
glPointSize(4.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}
void drawPolygonRotation(double angleRad)
{
glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 1.0);
```

```
for (int i = 0; i < edges; i++)
{
glVertex2i(round((pntX[i] * cos(angleRad)) - (pntY[i] *sin(angleRad))),
round((pntX[i]* sin(angleRad)) + (pntY[i] *cos(angleRad))));
}
glEnd();
}
void myDisplay(void)
{
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (0.0, 0.0, 0.0);
if (choice == 1)
{
drawPolygon();
drawPolygonTrans(transX, transY);
}
else if (choice == 2)
{
drawPolygon();
drawPolygonScale(scaleX, scaleY);
}
else if (choice == 3)
{
drawPolygon();
drawPolygonRotation(angleRad);
}
glFlush ();
}
int main(int argc, char** argv)
{
int angle;
cout << "Enter your choice:\n\n" << endl;
cout << "1. Translation" << endl;
cout << "2. Scaling" << endl;cout << "3. Rotation" << endl;
cout << "4. Exit" << endl;
cin >> choice;
if (choice == 4) {
return 0;
}
cout << "\n\nFor Polygon:\n" << endl;
cout << "Enter no of edges: "; cin >> edges;
for (int i = 0; i < edges; i++)
{
cout << "Enter co-ordinates for vertex " << i + 1 << " : ";
cin >> pntX1 >> pntY1;
pntX.push_back(pntX1);
pntY.push_back(pntY1);
}
if (choice == 1)
{
cout << "Enter the translation factor for X and Y: ";
cin >> transX >> transY;
}
else if (choice == 2)
{
cout << "Enter the scaling factor for X and Y: ";
cin >> scaleY >> scaleX;
}
```

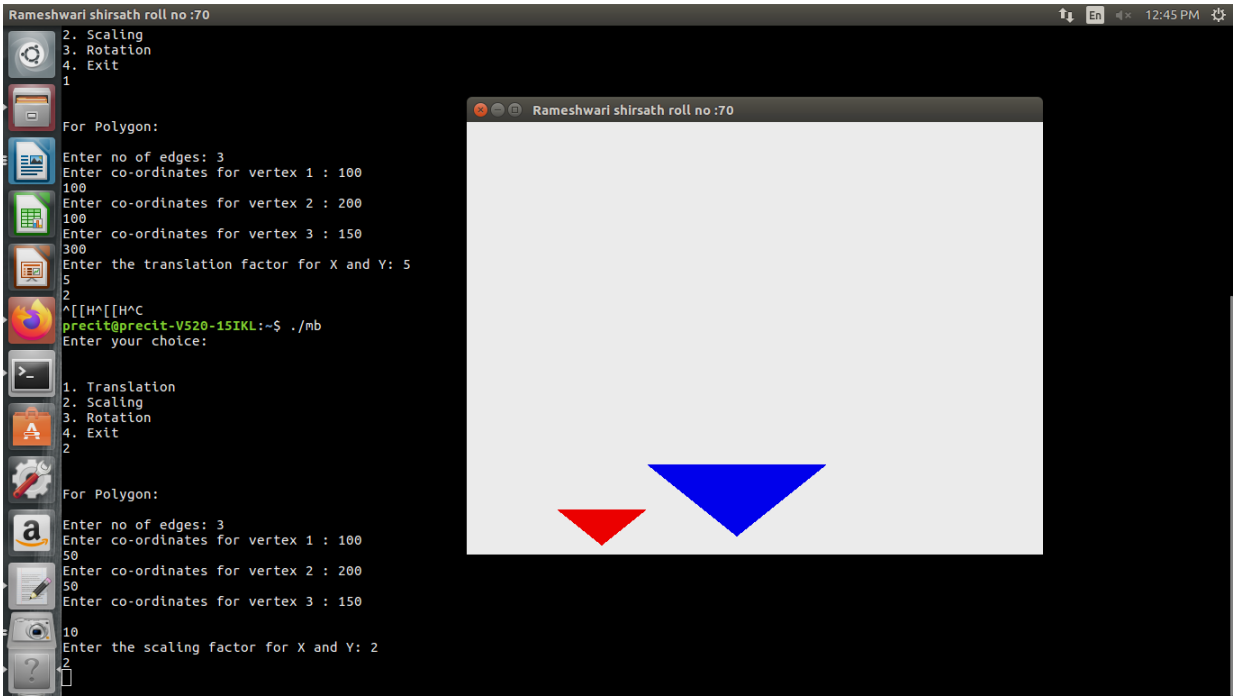
```
else if (choice == 3)
{
cout<<"Enter the angle of rotation:";
cin>>angle;
angleRad = angle * 3.1416 / 180;
}
//cout << "\n\nPoints:" << pntX[0] << ", " << pntY[0] << endl;
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (640, 480);
glutInitWindowPosition (100, 150);
glutCreateWindow ("Rameshwari Shirsath Roll No:70");
glutDisplayFunc(myDisplay);
myInit ();
glutMainLoop();
}
```

OUTPUT:

TRANSLATION:



SCALING:



ROTATION: