**Name – Pawar Shrikant Sanjeev**

**SE IT**

**Roll No.47**

**CG Lab**

**Assignment No. 6 -2D Transformations**

**#include <math.h>**

**#include <time.h>**

**#include <GL/freeglut.h>**

**#include<iostream>**

**using namespace std;**

**// this creates delay between two actions**

**void delay(unsigned int mseconds)**

**{**

**clock\_t goal = mseconds + clock();**

**while (goal > clock())**

**;**

**}**

**// this is a basic init for the glut window**

**void myInit(void)**

**{**

**glClearColor(1.0, 1.0, 1.0, 0.0);**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**gluOrtho2D(0.0, 640, 0.0, 480);**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glFlush();**

**}**

**// this function just draws a point**

**void drawRect(int a, int b)**

**{**

**glPointSize(1.0);**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**//creating a rectangle**

**glBegin(GL\_LINE\_LOOP);**

**glVertex2f(a,b);**

**glVertex2f(a,a);**

**glVertex2f(b,a);**

**glVertex2f(b,b);**

**glEnd();**

**glFlush();**

**}**

**void rotateAroundPt(int px, int py, int cx, int cy)**

**{**

**float theta = 45.0;**

**float xf1,xf2,yf1,yf2,xf3,yf3,xf4,yf4;**

**glColor3f(0.0f, 0.0f, 1.0f);**

**drawRect(px, py); // drawing the point**

**delay(1000);**

**theta=theta\*3.14/180.0;**

**// actual calculations..**

**xf1=cx + (int)((float)(px - cx) \* cos(theta))**

**- ((float)(py - cy) \* sin(theta));**

**yf1= cy + (int)((float)(px - cx) \* sin(theta))**

**+ ((float)(py - cy) \* cos(theta));**

**xf2 = cx + (int)((float)(px - cx) \* cos(theta))**

**- ((float)(px - cy) \* sin(theta));**

**yf2 = cy + (int)((float)(px - cx) \* sin(theta))**

**+ ((float)(px - cy) \* cos(theta));**

**xf3 = cx + (int)((float)(py - cx) \* cos(theta))**

**- ((float)(px - cy) \* sin(theta));**

**yf3= cy + (int)((float)(py - cx) \* sin(theta))**

**+ ((float)(px - cy) \* cos(theta));**

**xf4 = cx + (int)((float)(py - cx) \* cos(theta))**

**- ((float)(py - cy) \* sin(theta));**

**yf4= cy + (int)((float)(py - cx) \* sin(theta))**

**+ ((float)(py - cy) \* cos(theta));**

**glColor3f(1.0f, 0.0f, 0.0f);**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glColor3f(1.0f,0.0f,0.0f);**

**glBegin(GL\_LINE\_LOOP);**

**glVertex2f(xf1,yf1);**

**glVertex2f(xf2,yf2);**

**glVertex2f(xf3,yf3);**

**glVertex2f(xf4,yf4);**

**glEnd();**

**glFlush();**

**}**

**// this function will translate the point**

**void translatePoint(int px, int py, int tx, int ty)**

**{**

**int fx = px, fy = py;**

**glColor3f(0.0f, 0.0f, 1.0f);**

**drawRect(px, py); // drawing the point**

**delay(1000);**

**// update**

**px = px + tx;**

**py = py + ty;**

**glColor3f(1.0f, 0.0f, 0.0f);**

**drawRect(px, py); // drawing the point**

**glFlush();**

**}**

**// this function draws**

**void scalePoint(int px, int py, int sx, int sy)**

**{**

**int fx, fy;**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**// update**

**glColor3f(0.0f, 0.0f, 1.0f);**

**drawRect(px, py); // drawing the point**

**delay(1000);**

**fx = px \* sx;**

**fy = py \* sy;**

**glColor3f(1.0f, 0.0f, 0.0f);**

**drawRect(fx, fy); // drawing the point**

**delay(1000);**

**glFlush();**

**}**

**// Actual display function**

**void myDisplay(void)**

**{**

**int opt;**

**cout<<"\nEnter\n\t<1> for translation"**

**"\n\t<2> for rotation"**

**"\n\t<3> for scaling\n\t:";**

**cin>>opt;**

**cout<<"\nGo to the window...";**

**switch (opt) {**

**case 1:**

**translatePoint(100, 200, 50, 50);**

**break;**

**case 2:**

**rotateAroundPt(100, 200, 150, 150);**

**// point will circle around**

**// the centre of the window**

**break;**

**case 3:**

**scalePoint(100, 200, 2, 2);**

**break;**

**}**

**}**

**int main(int argc, char\*\* argv)**

**{**

**glutInit(&argc, argv);**

**glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);**

**glutInitWindowSize(640, 480);**

**glutInitWindowPosition(100, 150);**

**glutCreateWindow("Transforming point");**

**glutDisplayFunc(myDisplay);**

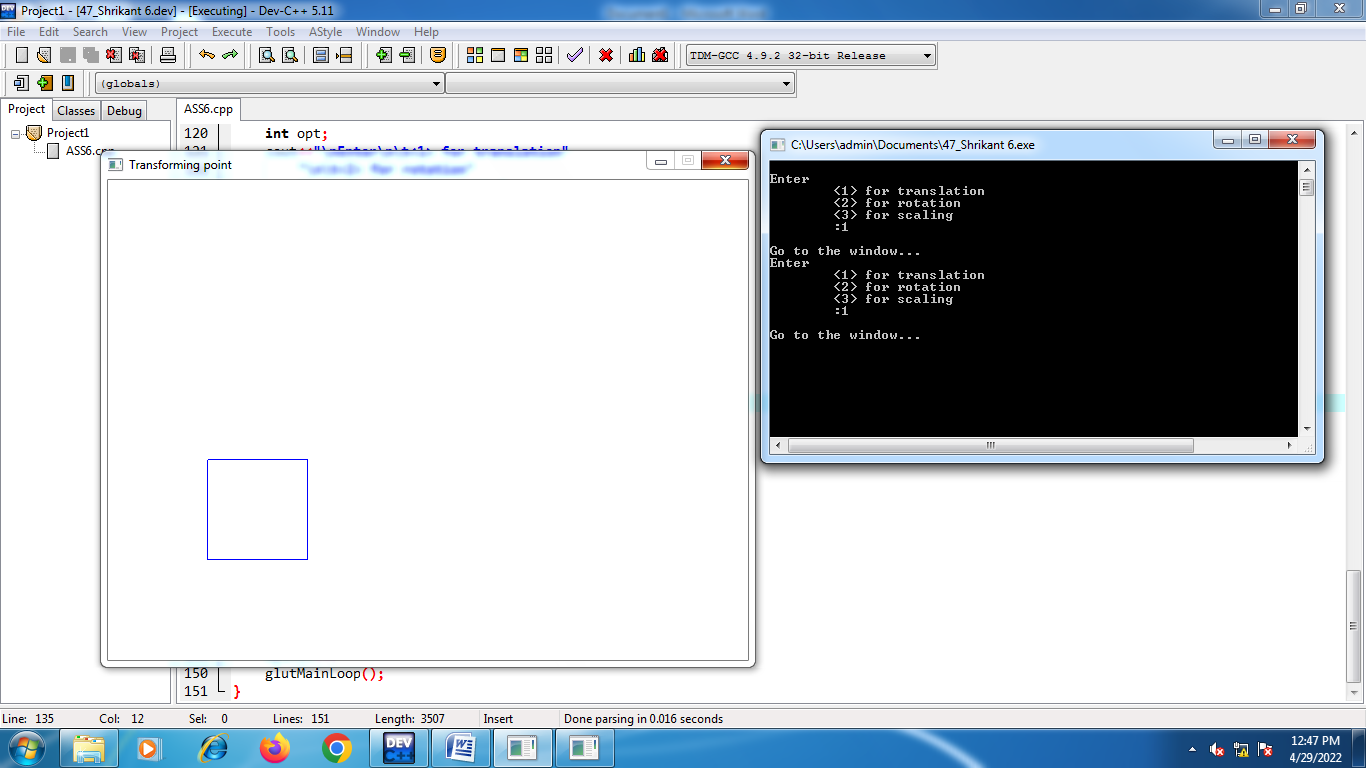
**myInit();**

**glutMainLoop();**

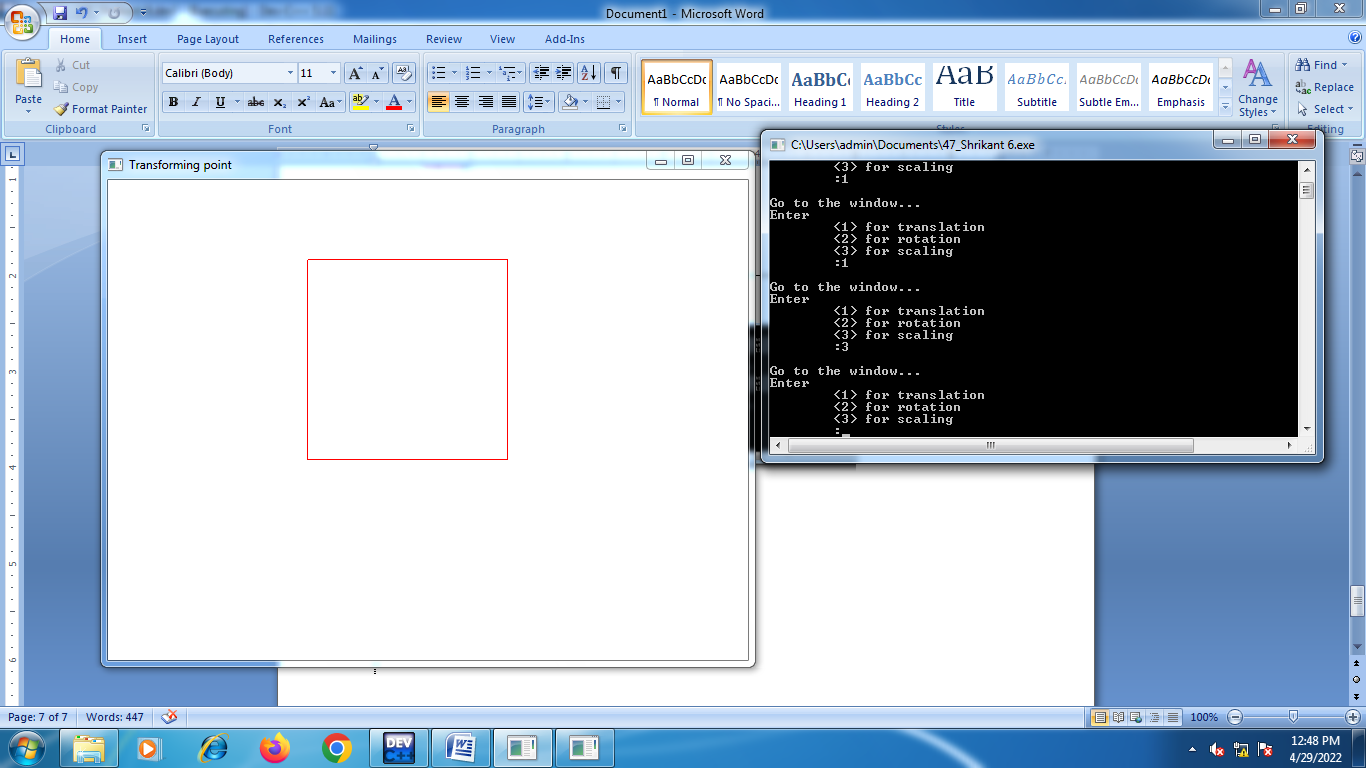
**}**

**OUTPUT:**

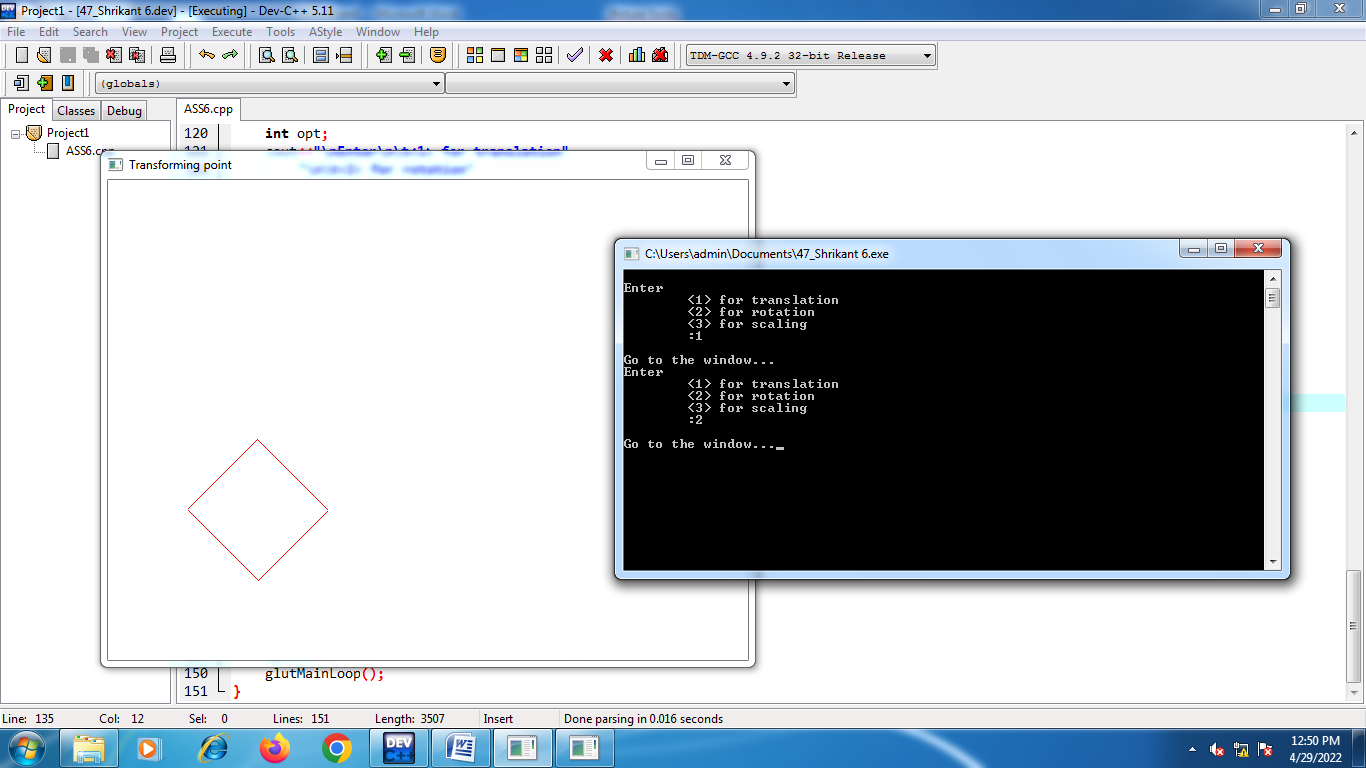
**Translation:**

****

**Scaling:**

****

**Rotation:**

****