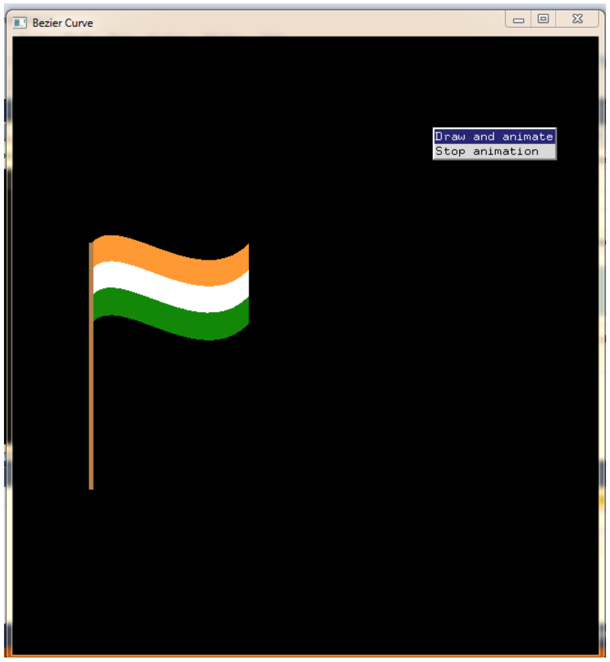
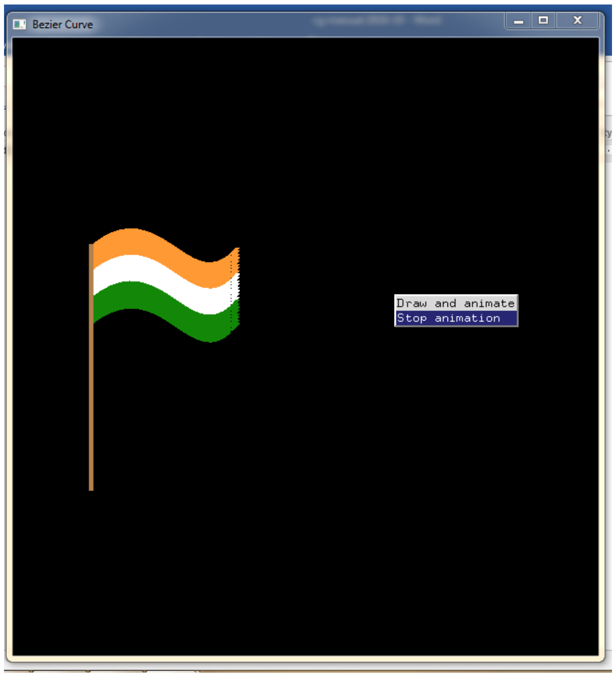
```
//8. Develop a menu driven program to animate a flag using Bezier Curve algorithm

#include<GL/glut.h>
#include<stdio.h>
#include<math.h>

#define pi 3.1416

static float th = 0;
GLint nCP = 4, nBCP = 20;

typedef struct wc
{
	GLfloat x, y, z;
};

void bino(GLint n, GLint *c)
{
	GLint k, j;
	for (k = 0; k <= n; k++)
	{
		c[k] = 1;
		for (j = n; j >= k + 1; j--)
			c[k] *= j;
		for (j = n - k; j >= 2; j--)
			c[k] /= j;
	}
}

void computeBezPt(GLfloat u, wc *bP, GLint nCP, wc *cP, GLint *c)
{
	GLint k, n = nCP - 1;
	GLfloat BEZ;

	bP->x = bP->y = bP->z = 0;
	for (k = 0; k<nCP; k++)
	{
		BEZ = c[k] * pow(u, k)*pow(1 - u, n - k);
		bP->x += cP[k].x*BEZ;
		bP->y += cP[k].y*BEZ;
		bP->z += cP[k].z*BEZ;
	}
}


void bezier(wc *cP, GLint nCP, GLint nBCP)
{
	wc bCP;
	GLfloat u;
	GLint *c, k;
	c = new GLint[nCP];
	bino(nCP - 1, c);
	glBegin(GL_LINE_STRIP);
	for (k = 0; k <= nBCP; k++)
	{
		u = GLfloat(k) / GLfloat(nBCP);
		computeBezPt(u, &bCP, nCP, cP, c);
		glVertex2f(bCP.x, bCP.y);
```

```
        }
        glEnd();
        delete[]c;
}

void display()
{
        glClearColor(0, 0, 0, 1);
}

void draw_and_animate()
{
        wc cP[4] = { { 20,100,0 },{ 30,110,0 },{ 50,90,0 },{ 60,100,0 } };

        cP[1].x += 10 * sin(th*pi / 180);
        cP[1].y += 5 * sin(th*pi / 180);

        cP[2].x -= 10 * sin((th + 30)*pi / 180);
        cP[2].y -= 10 * sin((th + 30)*pi / 180);

        cP[3].x -= 4 * sin(th*pi / 180);
        cP[3].x += sin((th - 30)*pi / 180);

        th += 0.1;

        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1, 1, 1);
        glPushMatrix();
        glLineWidth(5);

        glColor3f(255 / 255, 153 / 255.0, 51 / 255.0); //saffron
        for (int i = 0; i<8; i++)
        {
                glTranslatef(0, -.8, 0);
                bezier(cP, nCP, nBCP);
        }

        glColor3f(1, 1, 1); //white
        for (int i = 0; i<8; i++)
        {
                glTranslatef(0, -.8, 0);
                bezier(cP, nCP, nBCP);
        }

        glColor3f(19 / 255.0, 136 / 255.0, 8 / 255.0); //green
        for (int i = 0; i<8; i++)
        {
                glTranslatef(0, -.8, 0);
                bezier(cP, nCP, nBCP);
        }

        glPopMatrix();

        glColor3f(.7, .5, .3); //flag pole
        glLineWidth(5);
        glBegin(GL_LINES);
        glVertex2f(20, 100);
```

```c
	glVertex2f(20, 40);
	glEnd();
	glFlush();

	glutPostRedisplay();
	glutSwapBuffers();
}

void reshape(GLint w, GLint h)
{
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0, 150, 0, 150);
	glClear(GL_COLOR_BUFFER_BIT);
}



void menu(int id)
{
	switch (id)
	{
	case 1:glutIdleFunc(draw_and_animate);
		break;

	case 2:glutIdleFunc(NULL);
		break;
	}
	glutPostRedisplay();
}

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

	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
	glutInitWindowPosition(50, 50);
	glutInitWindowSize(640, 840);
	glutCreateWindow("Bezier Curve");
	glutReshapeFunc(reshape);
	glutDisplayFunc(display);
	glClearColor(0, 0, 0, 1);
	glFlush();
	glutCreateMenu(menu);
	glutAddMenuEntry("Draw and animate", 1);
	glutAddMenuEntry("Stop animation", 2);
	glutAttachMenu(GLUT_LEFT_BUTTON);
	glutMainLoop();
	return 0;
}
```

Output


Bezier Curve
Draw and animate
Stop animation


Bezier Curve
Draw and animate
Stop animation