```
#include<GL/glut.h>
#include<stdio.h>
#include<math.h>
#define PI 3.1416

float theta = 0;

struct point
{
    GLfloat x, y, z;
};

int factorial(int n)
{
    if (n<=1)
        return(1);
    else
        n=n*factorial(n-1);
    return n;
}

void computeNcR(int n, int *hold_ncr_values)
{
    int r;
    for(r=0; r<=n; r++) //start from nC0, then nC1, nC2, nC3 till nCn
    {
        hold_ncr_values[r] = factorial(n) / (factorial(n-r) * factorial(r));
    }
}

void computeBezierPoints(float t, point *actual_bezier_point, int
number_of_control_points, point *control_points_array, int *hold_ncr_values)
{
    int i, n = number_of_control_points-1;

    float bernstein_polynomial;

    actual_bezier_point  ->  x  =
    0; actual_bezier_point  -> y
    =  0; actual_bezier_point  ->
    z = 0;

    for(i=0; i < number_of_control_points; i++)
    {
        bernstein_polynomial = hold_ncr_values[i] * pow(t, i) * pow( 1-t, n-i);

        actual_bezier_point     ->     x     +=     bernstein_polynomial     *
        control_points_array[i].x; actual_bezier_point -> y += bernstein_polynomial
        *     control_points_array[i].y;     actual_bezier_point     ->     z     +=
        bernstein_polynomial * control_points_array[i].z;
    }
}

void bezier(point *control_points_array, int number_of_control_points, int
number_of_bezier_points)
{
    point actual_bezier_point;
    float t;
    int *hold_ncr_values, i;
    hold_ncr_values = new int[number_of_control_points]; // to hold the nCr values
    computeNcR(number_of_control_points-1, hold_ncr_values); // call nCr function
    to
calculate nCr values

    glBegin(GL_LINE_STRIP);
        for(i=0; i<=number_of_bezier_points; i++)
        {
            t=float(i)/float(number_of_bezier_points);
            computeBezierPoints(t, &actual_bezier_point, number_of_control_points,
control_points_array, hold_ncr_values);
            glVertex2f(actual_bezier_point.x, actual_bezier_point.y);
        }
    glEnd();
    delete[] hold_ncr_values;
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    int number_of_control_points = 4, number_of_bezier_points = 20;

    point control_points_array[4] = {{100, 400, 0}, {150, 450, 0}, {250, 350, 0},{300,
400, 0}};

    control_points_array[1].x += 50*sin(theta *
    PI/180.0); control_points_array[1].y += 25*sin(theta
    * PI/180.0);

    control_points_array[2].x -= 50*sin((theta+30) *
    PI/180.0); control_points_array[2].y -= 50*sin((theta+30)
    * PI/180.0);

    control_points_array[3].x -= 25*sin((theta-30) *
    PI/180.0); control_points_array[3].y += sin((theta-30) *
    PI/180.0);

    theta += 2;

    glPushMatrix();

    glPointSize(5);

    glColor3f(1, 0.4, 0.2); //Indian flag: Saffron color code
    for(int i=0; i<50; i++)
    {
        glTranslatef(0, -0.8, 0 );
        bezier(control_points_array, number_of_control_points, number_of_bezier_points);
    }

    glColor3f(1, 1, 1); //Indian flag: white color code
    for(int i=0; i<50; i++)
    {
        glTranslatef(0, -0.8, 0);
        bezier(control_points_array, number_of_control_points, number_of_bezier_points);
    }

    glColor3f(0, 1, 0); //Indian flag: green color code
    for(int i=0; i<50; i++)
    {
        glTranslatef(0, -0.8, 0);
        bezier(control_points_array, number_of_control_points, number_of_bezier_points);
    }

    glPopMatrix();

    glLineWidth(5);

    glColor3f(0.7, 0.5,0.3); //pole colour
    glBegin(GL_LINES);
        glVertex2f(100,400);
        glVertex2f(100,40);
    glEnd();

    glutPostRedisplay();
    glutSwapBuffers();
}

void init()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,500,0,500);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(500,500);
    glutCreateWindow("Bezier Curve - updated");

    init();

    glutDisplayFunc(display);

    glutMainLoop();
}
```