```c
#include <GL/glew.h>
#include <GL/glut.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define my_PI 3.141592

static char* vsSource = "#version 120 ₩n₩
in vec4 aPosition; ₩n₩
in vec4 aColor; ₩n₩
out vec4 vColor; ₩n₩
uniform mat4 urotate;  ₩n₩
void main(void) { ₩n₩
  gl_Position = urotate*aPosition; ₩n₩
  vColor = aColor; ₩n₩
}";

static char* fsSource = "#version 120 ₩n₩
in vec4 vColor; ₩n₩
void main(void) { ₩n₩
  gl_FragColor = vColor; ₩n₩
}";

GLuint vs = 0;
GLuint fs = 0;
GLuint prog = 0;

char buf[1024];
float t = 0.0f;

GLfloat vertices[] = {
    -0.2, -0.2, -0.2, 1.0, // 0
    -0.2, -0.2, +0.2, 1.0, // 1
    -0.2, +0.2, -0.2, 1.0, // 2
    -0.2, +0.2, +0.2, 1.0, // 3
    +0.2, -0.2, -0.2, 1.0, // 4
    +0.2, -0.2, +0.2, 1.0, // 5
    +0.2, +0.2, -0.2, 1.0, // 6
    +0.2, +0.2, +0.2, 1.0, // 7
};


GLfloat colors[] = {
    1.0, 0.0, 0.0, 1.0,
    0.0, 1.0, 0.0, 1.0,
    0.0, 0.0, 1.0, 1.0,
    1.0, 1.0, 0.0, 1.0,
    0.0, 1.0, 1.0, 1.0,
    1.0, 0.0, 1.0, 1.0,
    1.0, 0.5, 0.2, 1.0,
    0.2, 1.0, 1.0, 1.0,
};

GLushort indices[] = { // 36 points, 12 triangles
    0, 4, 6,
    6, 2, 0,
    4, 5, 7,
    7, 6, 4,
    1, 3, 7,
    7, 5, 1,
    0, 2, 3,
    3, 1, 0,
    2, 6, 7,
    7, 3, 2,
    0, 1, 5,
    5, 4, 0,
};

void myinit(void) {
    GLuint status;

    printf("***** Your student number and name *****₩n");
    vs = glCreateShader(GL_VERTEX_SHADER);
    glShaderSource(vs, 1, &vsSource, NULL);
    glCompileShader(vs);
    glGetShaderiv(vs, GL_COMPILE_STATUS, &status);
    printf("vs compile status = %s₩n", (status == GL_TRUE) ? "true" : 
      "false");
    glGetShaderInfoLog(vs, sizeof(buf), NULL, buf);
    printf("vs log = [%s]₩n", buf);

    fs = glCreateShader(GL_FRAGMENT_SHADER);
    glShaderSource(fs, 1, &fsSource, NULL);
    glCompileShader(fs);
    glGetShaderiv(fs, GL_COMPILE_STATUS, &status);
    printf("fs compile status = %s₩n", (status == GL_TRUE) ? "true" : 
      "false");
    glGetShaderInfoLog(fs, sizeof(buf), NULL, buf);
    printf("fs log = [%s]₩n", buf);

    prog = glCreateProgram();
    glAttachShader(prog, vs);
    glAttachShader(prog, fs);
    glLinkProgram(prog);
    glGetProgramiv(prog, GL_LINK_STATUS, &status);
    printf("program link status = %s₩n", (status == GL_TRUE) ? "true" : 
      "false");
    glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
    printf("link log = [%s]₩n", buf);
    glValidateProgram(prog);
    glGetProgramiv(prog, GL_VALIDATE_STATUS, &status);
    printf("program validate status = %s₩n", (status == GL_TRUE) ? "true" : 
      "false");
    glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
    printf("validate log = [%s]₩n", buf);
    glUseProgram(prog);

    GLuint loc;
    GLuint vbo[1];
    // using vertex buffer object
    glGenBuffers(1, vbo);
```

```c
    glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
    glBufferData(GL_ARRAY_BUFFER, 2 * 8 * 4 * sizeof(GLfloat), NULL,
      GL_STATIC_DRAW);
    glBufferSubData(GL_ARRAY_BUFFER, 0, 8 * 4 * sizeof(GLfloat), vertices);
    glBufferSubData(GL_ARRAY_BUFFER, 8 * 4 * sizeof(GLfloat), 8 * 4 * sizeof
      (GLfloat),
        colors);

    loc = glGetAttribLocation(prog, "aPosition");
    glEnableVertexAttribArray(loc);
    glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)0);

    loc = glGetAttribLocation(prog, "aColor");
    glEnableVertexAttribArray(loc);
    glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)(3 * 4 *
      sizeof(GLfloat)));

}

void mykeyboard(unsigned char key, int x, int y) {
    switch (key) {
    case 27: // ESCAPE
        exit(0);
        break;
    }
}



void myidle(void) {
    // redisplay
    glutPostRedisplay();
}

GLfloat m[16];

void mydisplay(void) {
    GLuint loc;
    glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
    glClear(GL_COLOR_BUFFER_BIT);

    t = 30.0 * my_PI/180.0;

    /* rotation about z-axis
    m[0] = cos(t); m[4] = -sin(t); m[8] = 0.0;  m[12] = 0.0;
    m[1] = sin(t); m[5] = cos(t);  m[9] = 0.0;  m[13] = 0.0;
    m[2] = 0.0;    m[6] = 0.0;  m[10] = 1.0; m[14] = 0.0;
    m[3] = 0.0;    m[7] = 0.0;     m[11] = 0.0; m[15] = 1.0;
    */
    // rotation about x-axis
    m[0] = 1.0; m[4] = 0.0; m[8] = 0.0;  m[12] = 0.0;
    m[1] = 0.0; m[5] = cos(t);  m[9] = -sin(t);  m[13] = 0.0;
    m[2] = 0.0;    m[6] = sin(t);  m[10] = cos(t); m[14] = 0.0;
    m[3] = 0.0;    m[7] = 0.0;     m[11] = 0.0; m[15] = 1.0;

    loc = glGetUniformLocation(prog, "urotate");
    glUniformMatrix4fv(loc, 1, GL_FALSE, m);
    glDrawElements(GL_TRIANGLES, 12 * 3, GL_UNSIGNED_SHORT, indices);
    glFlush();
    glutSwapBuffers();
}


int main(int argc, char* argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("*** Your Student Number and Name ***");
    glutDisplayFunc(mydisplay);
    glutIdleFunc(myidle);
    glutKeyboardFunc(mykeyboard);
    glewInit();
    myinit();
    glutMainLoop();
    return 0;
}
```