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

static char* vsSource = "#version 120 \n\
in vec4 aPosition; \n\
in vec4 aColor; \n\
out vec4 vColor; \n\
void main(void) { \n\
  gl_Position = 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];

void myinit(void) {
    GLuint status;
    printf("***** Your student number and name *****\n");
    // vs: vertex shader
    vs = glCreateShader(GL_VERTEX_SHADER);
    glShaderSource(vs, 1, &vsSource, NULL);
    glCompileShader(vs); // compile to get .OBJ
    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: fragment shader
    fs = glCreateShader(GL_FRAGMENT_SHADER);
    glShaderSource(fs, 1, &fsSource, NULL);
    glCompileShader(fs); // compile to get .OBJ
    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: program
    prog = glCreateProgram();
    glAttachShader(prog, vs);
    glAttachShader(prog, fs);
    glLinkProgram(prog); // link to get .EXE
    glGetProgramiv(prog, GL_LINK_STATUS, &status);
    printf("program link status = %s\n", (status == GL_TRUE) ? "true" :
      "false");
    glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
```

```
54      printf("link log = [%s]\n", buf);
55      glValidateProgram(prog);
56      glGetProgramiv(prog, GL_VALIDATE_STATUS, &status);
57      printf("program validate status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
58      glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
59      printf("validate log = [%s]\n", buf);
60      glUseProgram(prog); // execute it !
61  }
62
63  void mykeyboard(unsigned char key, int x, int y) {
64      switch (key) {
65      case 27: // ESCAPE
66          exit(0);
67          break;
68      }
69  }
70
71  GLfloat vertices[] = {
72      -0.5, -0.5, 0.0, 1.0,
73      +0.5, -0.5, 0.0, 1.0,
74      -0.5, +0.5, 0.0, 1.0,
75  };
76
77  GLfloat colors[] = {
78      1.0, 0.0, 0.0, 1.0, // red
79      0.0, 1.0, 0.0, 1.0, // green
80      0.0, 0.0, 1.0, 1.0, // blue
81  };
82
83  void mydisplay(void) {
84      GLuint loc;
85
86      glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
87      glClear(GL_COLOR_BUFFER_BIT);
88
89      loc = glGetAttribLocation(prog, "aPosition");
90      glEnableVertexAttribArray(loc);
91      glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, vertices);
92
93      loc = glGetAttribLocation(prog, "aColor");
94      glEnableVertexAttribArray(loc);
95      glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, colors);
96
97      glDrawArrays(GL_TRIANGLES, 0, 3);
98
99      glFlush();
100 }
101
102 int main(int argc, char* argv[]) {
103     glutInit(&argc, argv);
104     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
105     glutInitWindowSize(500, 500);
106     glutInitWindowPosition(0, 0);
107     glutCreateWindow("*** Your Student number and name ***");
108     glutDisplayFunc(mydisplay);
```

```
109     glutKeyboardFunc(mykeyboard);
110     glewInit();
111     myinit();
112     glutMainLoop();
113     return 0;
114 }
115
```