Lab Taks-4

Submission Guidelines-

- Rename the file to your id only. If your id is 18-XXXXX-1, then the file name must be 18-XXXXX-1.docx.
- Must submit within time that will be discussed in class VUES to the section named Lab Tak-4
- Must include resources for all the section in the table

Question- 1

Draw the scenario of a traffic signal

Graph Plot (Picture)-

(Not Needed)

Code-

```
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h


/* Initialize OpenGL Graphics */
void initGL() {
        // Set "clearing" or background color
        glClearColor(0.0f, 1.0f, 0.0f, 1.0f); // Black and opaque
}


void trafficSignal()
{
   glBegin(GL_POLYGON);         // These vertices form a closed polygon
        glColor3ub(93, 61, 9 );
        glVertex2f(-0.1f, -0.2f);
        glVertex2f(0.0f, -0.2f);
        glVertex2f(0.0f, -0.1f);
  glVertex2f(-0.1f, -0.1f);
  glEnd();

  glLineWidth(24);
```

```
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(125, 91, 36  );
    glVertex2f(-0.05f, -0.1f);
    glVertex2f(-0.05f, 0.2f);
    glEnd();

    glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(80, 64, 38  );
        glVertex2f(-0.1f, 0.2f);
        glVertex2f(0.0f, 0.2f);
        glVertex2f(0.0f, 0.6f);
    glVertex2f(-0.1f, 0.6f);
    glEnd();

    glLineWidth(6);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(0, 0, 0 );
    glVertex2f(-0.1f, 0.2f);
    glVertex2f(0.0f, 0.2f);
    glVertex2f(0.0f, 0.2f);
    glVertex2f(0.0f, 0.6f);
    glVertex2f(0.0f, 0.6f);
    glVertex2f(-0.1f, 0.6f);
    glVertex2f(-0.1f, 0.6f);
    glVertex2f(-0.1f, 0.2f);
    glEnd();

     glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(229, 255, 0);
        glVertex2f(-0.08f, 0.23f);
        glVertex2f(-0.02f, 0.23f);
        glVertex2f(-0.02f, 0.3f);
    glVertex2f(-0.08f, 0.3f);
    glEnd();

    glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(4, 117, 0 );
        glVertex2f(-0.08f, 0.35f);
        glVertex2f(-0.02f, 0.35f);
        glVertex2f(-0.02f, 0.43f);
    glVertex2f(-0.08f, 0.43f);
    glEnd();
```

```
   glBegin(GL_POLYGON);          // These vertices form a closed polygon
         glColor3ub(255, 4, 0 );
         glVertex2f(-0.08f, 0.47f);
         glVertex2f(-0.02f, 0.47f);
         glVertex2f(-0.02f, 0.55f);
   glVertex2f(-0.08f, 0.55f);
   glEnd();

}

void scenario()
{
   glBegin(GL_POLYGON);          // These vertices form a closed polygon
         glColor3ub(0, 0, 0 );
         glVertex2f(-0.99f, -0.7f);
         glVertex2f(0.99f, -0.7f);
         glVertex2f(0.99f, -0.10f);
   glVertex2f(-0.99f, -0.10f);
   glEnd();

   glBegin(GL_POLYGON);          // These vertices form a closed polygon
         glColor3ub(0, 0, 0 );
         glVertex2f(-0.2f, -0.7f);
         glVertex2f(0.4f, 0.9f);
         glVertex2f(0.8f, 0.9f);
   glVertex2f(0.4f, -0.7f);
   glEnd();

   glLineWidth(14);
         // Draw a Red 1x1 Square centered at origin
         glBegin(GL_LINES); // Each set of 4 vertices form a quad
         glColor3ub(255, 255, 255 );
   glVertex2f(-0.99f, -0.4f);
   glVertex2f(-0.98f, -0.4f);
   glVertex2f(-0.95f, -0.4f);
   glVertex2f(-0.88f, -0.4f);
   glVertex2f(-0.85f, -0.4f);
   glVertex2f(-0.78f, -0.4f);
   glVertex2f(-0.75f, -0.4f);
   glVertex2f(-0.68f, -0.4f);
   glVertex2f(-0.65f, -0.4f);
   glVertex2f(-0.58f, -0.4f);
   glVertex2f(-0.55f, -0.4f);
```

```
   glVertex2f(-0.48f, -0.4f);
   glVertex2f(-0.45f, -0.4f);
   glVertex2f(-0.38f, -0.4f);
   glVertex2f(-0.35f, -0.4f);
   glVertex2f(-0.28f, -0.4f);
   glVertex2f(-0.25f, -0.4f);
   glVertex2f(-0.15f, -0.4f);
   glVertex2f(-0.10f, -0.4f);
   glVertex2f(0.0f, -0.4f);
   glVertex2f(0.07f, -0.4f);
   glVertex2f(0.16f, -0.4f);
   glVertex2f(0.20f, -0.4f);
   glVertex2f(0.29f, -0.4f);
   glVertex2f(0.33f, -0.4f);
   glVertex2f(0.40f, -0.4f);
   glVertex2f(0.43f, -0.4f);
   glVertex2f(0.50f, -0.4f);
   glVertex2f(0.53f, -0.4f);
   glVertex2f(0.60f, -0.4f);
   glVertex2f(0.63f, -0.4f);
   glVertex2f(0.70f, -0.4f);
   glVertex2f(0.73f, -0.4f);
   glVertex2f(0.80f, -0.4f);
   glVertex2f(0.83f, -0.4f);
   glVertex2f(0.90f, -0.4f);
   glVertex2f(0.93f, -0.4f);
   glVertex2f(0.99f, -0.4f);
   glVertex2f(0.29f, -0.2f);
   glVertex2f(0.35f, 0.0f);
   glVertex2f(0.38f, 0.1f);
   glVertex2f(0.44f, 0.3f);
   glVertex2f(0.46f, 0.4f);
   glVertex2f(0.52f, 0.6f);
   glVertex2f(0.54f, 0.7f);
  glVertex2f(0.60f, 0.9f);
   glEnd();

}
/* Handler for window-repaint event. Call back when the window first appears and
whenever the window needs to be re-painted. */
void display() {

        glClear(GL_COLOR_BUFFER_BIT);   // Clear the color buffer with current clearing
color
```

```
   scenario();
   trafficSignal();
        glFlush();  // Render now
}
/* Main function: GLUT runs as a console application starting at main()  */
int main(int argc, char** argv) {
        glutInit(&argc, argv);        // Initialize GLUT
        glutCreateWindow("Vertex, Primitive & Color");  // Create window with the given
title
        glutInitWindowSize(320, 320);   // Set the window's initial width & height
        glutDisplayFunc(display);      // Register callback handler for window re-paint event
        initGL();                  // Our own OpenGL initialization
        glutMainLoop();              // Enter the event-processing loop
        return 0;
}
```

Output Screenshot (Full Screen)-