Lab Taks-3

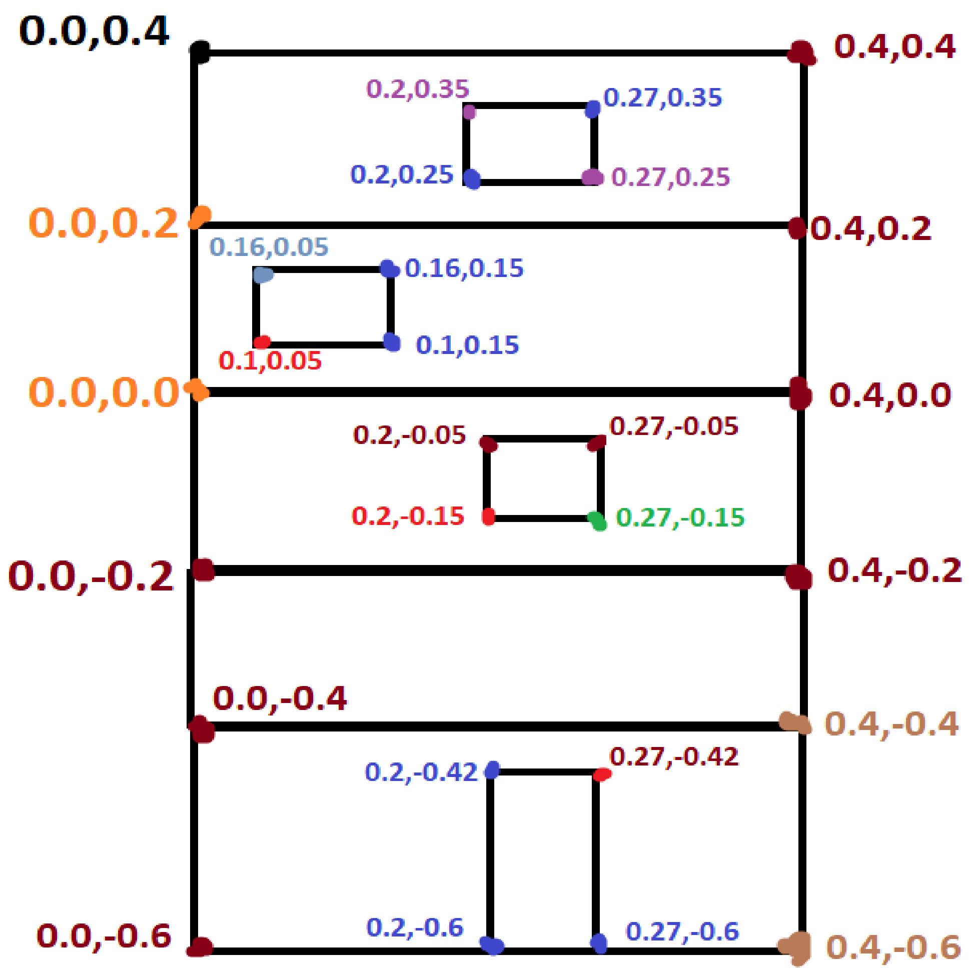
Submission Guidelines-

- 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-3
- Must include resources for all the section in the table

Question- 1

Draw five storied building with windows and a front door

Graph Plot (Picture)-



Code-

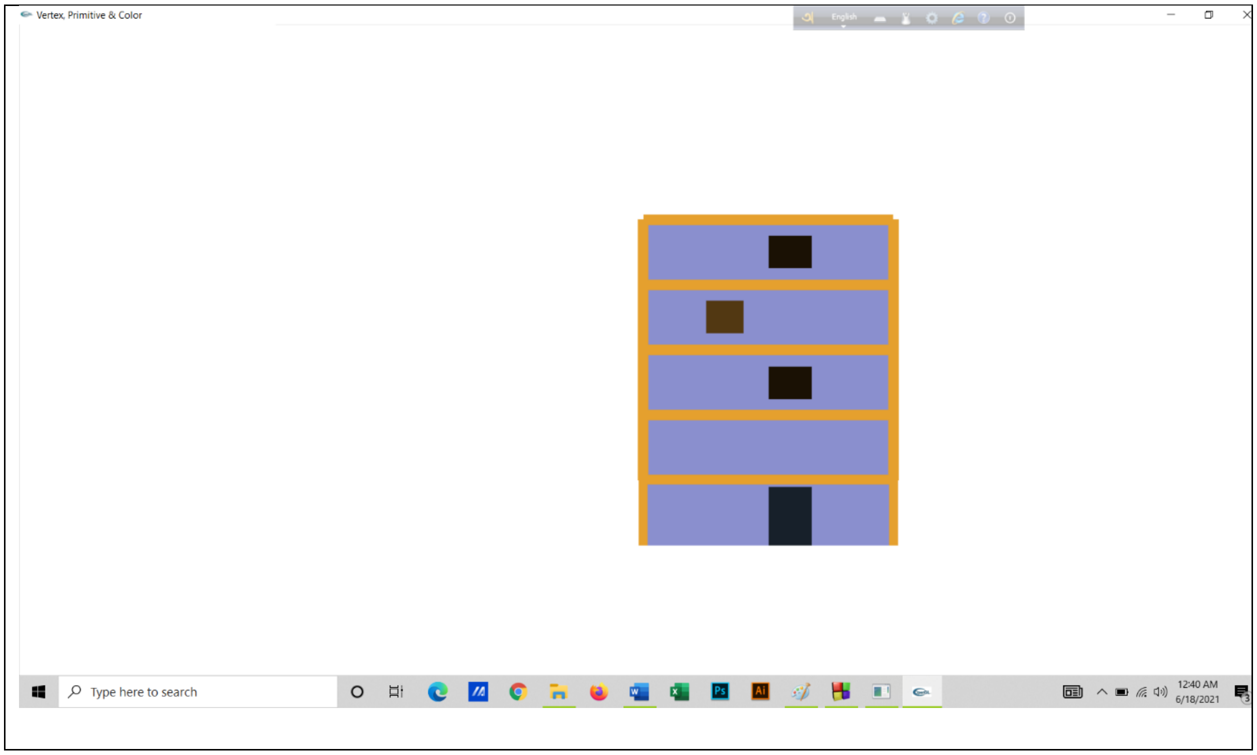
```
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h

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

/* 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
       glBegin(GL_POLYGON);           // These vertices form a closed polygon
	glColor3ub(138, 143, 206  );
	glVertex2f(0.0f, 0.0f);
	glVertex2f(0.0f, -0.2f);
	glVertex2f(0.4f, -0.2f);
   glVertex2f(0.4f, 0.0f);
   glEnd();

   glLineWidth(13);
	// Draw a Red 1x1 Square centered at origin
	glBegin(GL_LINES); // Each set of 4 vertices form a quad
	glColor3ub(229, 160, 46  );
	glVertex2f(0.0f, 0.0f);
	glVertex2f(0.0f, -0.2f);
   glVertex2f(0.0f, -0.2f);
   glVertex2f(0.4f, -0.2f);
   glVertex2f(0.4f, -0.2f);
	glVertex2f(0.4f, 0.0f);
	glVertex2f(0.4f, 0.0f);
	glVertex2f(0.0f, 0.0f);
   glEnd();
    glBegin(GL_POLYGON);           // These vertices form a closed polygon
	glColor3ub(138, 143, 206  );
	glVertex2f(0.0f, -0.2f);
	glVertex2f(0.0f, -0.4f);
	glVertex2f(0.4f, -0.4f);
   glVertex2f(0.4f, -0.2f);
   glEnd();
```

```
  glLineWidth(13);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(229, 160, 46 );
glVertex2f(0.0f, -0.2f);
        glVertex2f(0.0f, -0.4f);
glVertex2f(0.0f, -0.4f);
        glVertex2f(0.4f, -0.4f);
        glVertex2f(0.4f, -0.4f);
glVertex2f(0.4f, -0.2f);
glVertex2f(0.4f, -0.2f);
glVertex2f(0.0f, -0.2f);
glEnd();

 glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(138, 143, 206  );
        glVertex2f(0.0f, -0.4f);
        glVertex2f(0.0f, -0.6f);
        glVertex2f(0.4f, -0.6f);
glVertex2f(0.4f, -0.4f);
glEnd();

glLineWidth(11);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(229, 160, 46 );
        glVertex2f(0.0f, -0.4f);
        glVertex2f(0.0f, -0.6f);
        //glVertex2f(0.0f, -0.6f);
glVertex2f(0.4f, -0.6f);
//glVertex2f(0.4f, -0.6f);
glVertex2f(0.4f, -0.4f);
glVertex2f(0.4f, -0.4f);
        glVertex2f(0.0f, -0.4f);
 glEnd();

 glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(138, 143, 206  );
        glVertex2f(0.0f, 0.2f);
        glVertex2f(0.0f, 0.0f);
        glVertex2f(0.4f, 0.0f);
glVertex2f(0.4f, 0.2f);
glEnd();
```

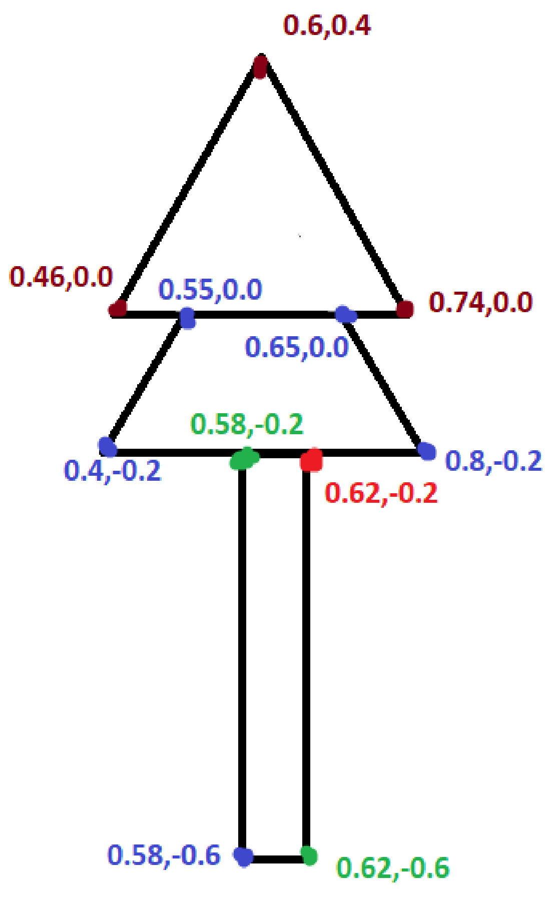
```
glLineWidth(13);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(229, 160, 46 );
        glVertex2f(0.0f, 0.2f);
        glVertex2f(0.0f, 0.0f);
glVertex2f(0.0f, 0.0f);
        glVertex2f(0.4f, 0.0f);
glVertex2f(0.4f, 0.0f);
glVertex2f(0.4f, 0.2f);
glVertex2f(0.4f, 0.2f);
        glVertex2f(0.0f, 0.2f);
glEnd();

 glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(138, 143, 206  );
        glVertex2f(0.0f, 0.4f);
        glVertex2f(0.0f, 0.2f);
        glVertex2f(0.4f, 0.2f);
glVertex2f(0.4f, 0.4f);
glEnd();

 glLineWidth(13);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(229, 160, 46  );
        glVertex2f(0.0f, 0.4f);
        glVertex2f(0.0f, 0.2f);
glVertex2f(0.0f, 0.2f);
        glVertex2f(0.4f, 0.2f);
glVertex2f(0.4f, 0.2f);
glVertex2f(0.4f, 0.4f);
glVertex2f(0.4f, 0.4f);
glVertex2f(0.0f, 0.4f);
glEnd();

  glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(23, 32, 42 );
        glVertex2f(0.2f, -0.42f);
        glVertex2f(0.2f, -0.6f);
        glVertex2f(0.27f, -0.6f);
glVertex2f(0.27, -0.42f);
glEnd();
```

```
  glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(26, 17, 4  );
        glVertex2f(0.2f, -0.15);
        glVertex2f(0.27f, -0.15f);
        glVertex2f(0.27f, -0.05f);
  glVertex2f(0.2, -0.05f);
  glEnd();


  glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(82, 56, 18  );
        glVertex2f(0.1f, 0.05);
        glVertex2f(0.16f, 0.05f);
        glVertex2f(0.16f, 0.15f);
  glVertex2f(0.1f, 0.15f);
  glEnd();

  glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(26, 17, 4  );
        glVertex2f(0.2f, 0.35);
        glVertex2f(0.2f, 0.25f);
        glVertex2f(0.27, 0.25f);
  glVertex2f(0.27f, 0.35f);
  glEnd();


        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)-



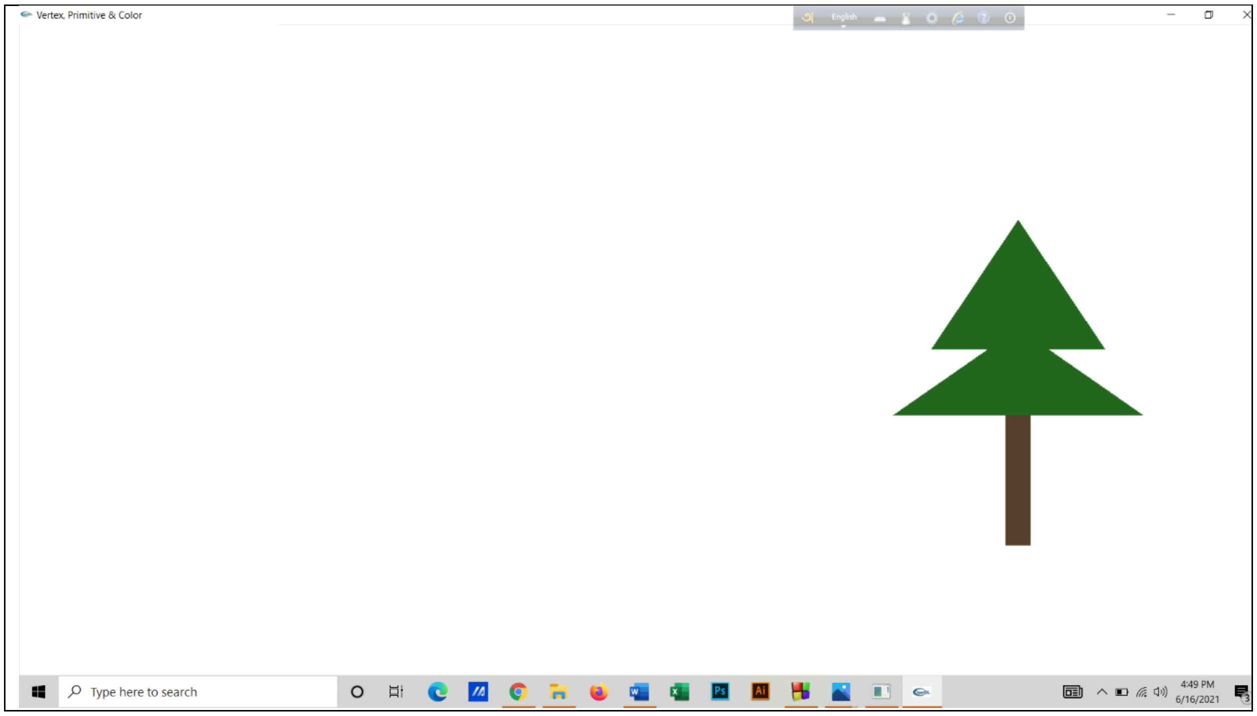
Question- 2 Draw a tree
Graph Plot (Picture)-



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(1.0f, 1.0f, 1.0f, 1.0f); // Black and opaque
}

/* 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
    glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(35, 103, 31  );
        glVertex2f(0.46f, 0.0f);
        glVertex2f(0.74f, 0.0f);
        glVertex2f(0.6f, 0.4f);
   glEnd();


    glBegin(GL_POLYGON);            // These vertices form a closed polygon
        glColor3ub(35, 103, 31  );
        glVertex2f(0.65f, 0.0f);
        glVertex2f(0.55f, 0.0f);
        glVertex2f(0.4f, -0.2f);
   glVertex2f(0.8f, -0.2f);
   glEnd();

   glBegin(GL_POLYGON);            // These vertices form a closed polygon
        glColor3ub(85, 62, 43   );
        glVertex2f(0.62f, -0.2f);
        glVertex2f(0.58f, -0.2f);
        glVertex2f(0.58f, -0.6f);
   glVertex2f(0.62f, -0.6f);
   glEnd();
        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;
}
```

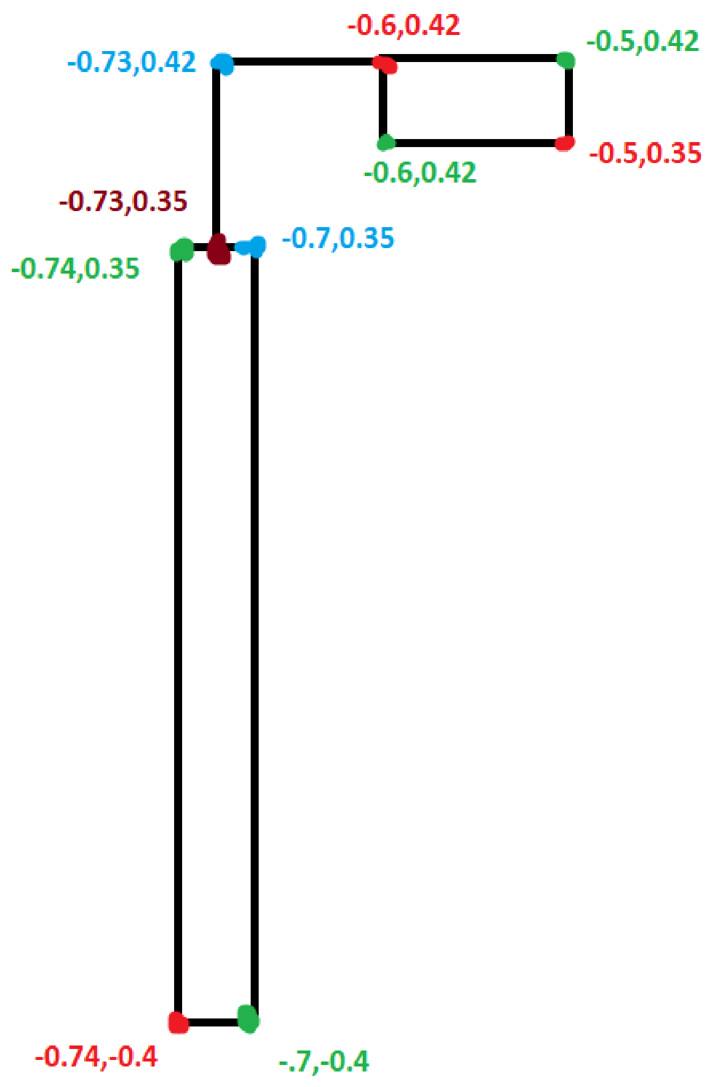
Output Screenshot (Full Screen)-



Question- 3

Draw a lamppost with black background

Graph Plot (Picture)-

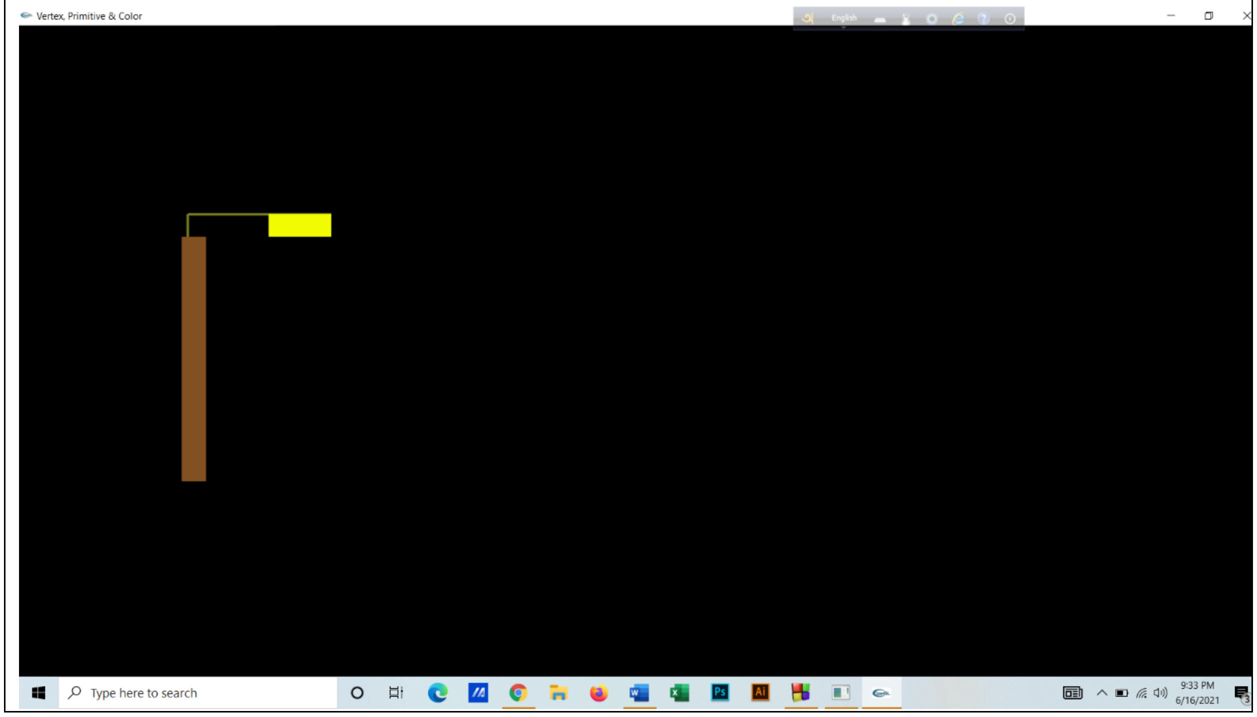


Code-

```
#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, 0.0f, 0.0f, 1.0f); // Black and opaque
}
```

```
/* 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
    glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(130, 80, 33  );
        glVertex2f(-0.74f, 0.35f);
        glVertex2f(-0.74f, -0.4f);
        glVertex2f(-0.7f, -0.4f);
   glVertex2f(-0.7f, 0.35f);
   glEnd();

   glLineWidth(3);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(121, 125, 32 );
   glVertex2f(-0.5f, 0.42f);
   glVertex2f(-0.73f, 0.42f);
   glVertex2f(-0.73f, 0.42f);
   glVertex2f(-0.73f, 0.35f);
   glEnd();

    glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(241, 252, 0 );
        glVertex2f(-0.5f, 0.42f);
        glVertex2f(-0.6f, 0.42f);
        glVertex2f(-0.6f, 0.35f);
   glVertex2f(-0.5f, 0.35f);
   glEnd();

        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;
}
```

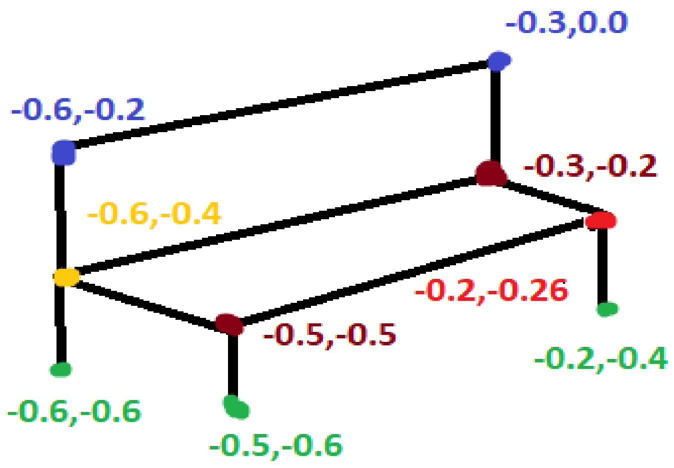
Output Screenshot (Full Screen)-



Question- 4

Draw a bench

Graph Plot (Picture)-

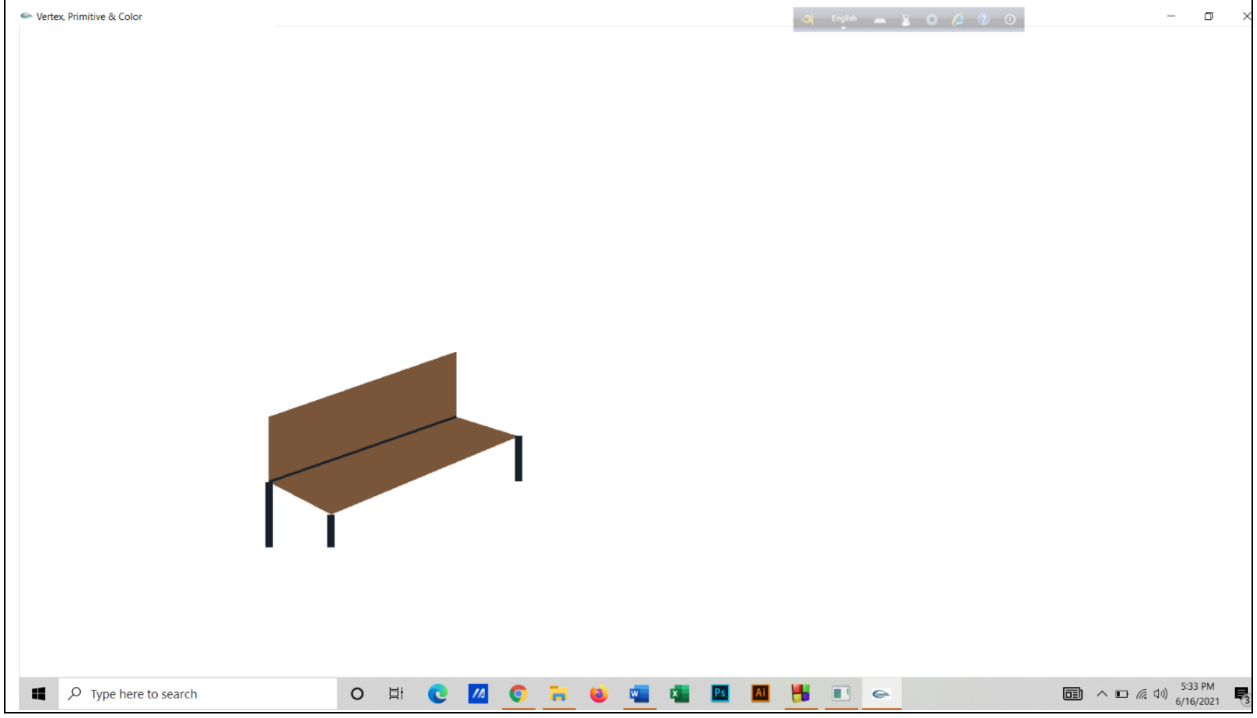


Code-

```
#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(1.0f, 1.0f, 1.0f, 1.0f); // Black and opaque
}

/* 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
    glBegin(GL_POLYGON);           // These vertices form a closed polygon
	glColor3ub(121, 85, 57  );
	glVertex2f(-0.3f, 0.0f);
	glVertex2f(-0.6f, -0.2f);
	glVertex2f(-0.6f, -0.4f);
   glVertex2f(-0.3f, -0.2f);
   glEnd();

    glBegin(GL_POLYGON);           // These vertices form a closed polygon
	glColor3ub(121, 85, 57  );
	glVertex2f(-0.3f, -0.2f);
	glVertex2f(-0.6f, -0.4f);
	glVertex2f(-0.5f, -0.5f);
   glVertex2f(-0.2f, -0.26f);
   glEnd();

   glLineWidth(3);
	// Draw a Red 1x1 Square centered at origin
	glBegin(GL_LINES); // Each set of 4 vertices form a quad
	glColor3ub(23, 32, 42 );
	glVertex2f(-0.3f, -0.2f);
	glVertex2f(-0.6f, -0.4f);
   glEnd();


   glLineWidth(9);
	// Draw a Red 1x1 Square centered at origin
	glBegin(GL_LINES); // Each set of 4 vertices form a quad
	glColor3ub(23, 32, 42 );
```

```
        glVertex2f(-0.6f, -0.4f);
        glVertex2f(-0.6f, -0.6f);
  glVertex2f(-0.2f, -0.26f);
        glVertex2f(-0.2f, -0.4f);
  glVertex2f(-0.5f, -0.5f);
  glVertex2f(-0.5f, -0.6f);

  glEnd();
        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;
}
```

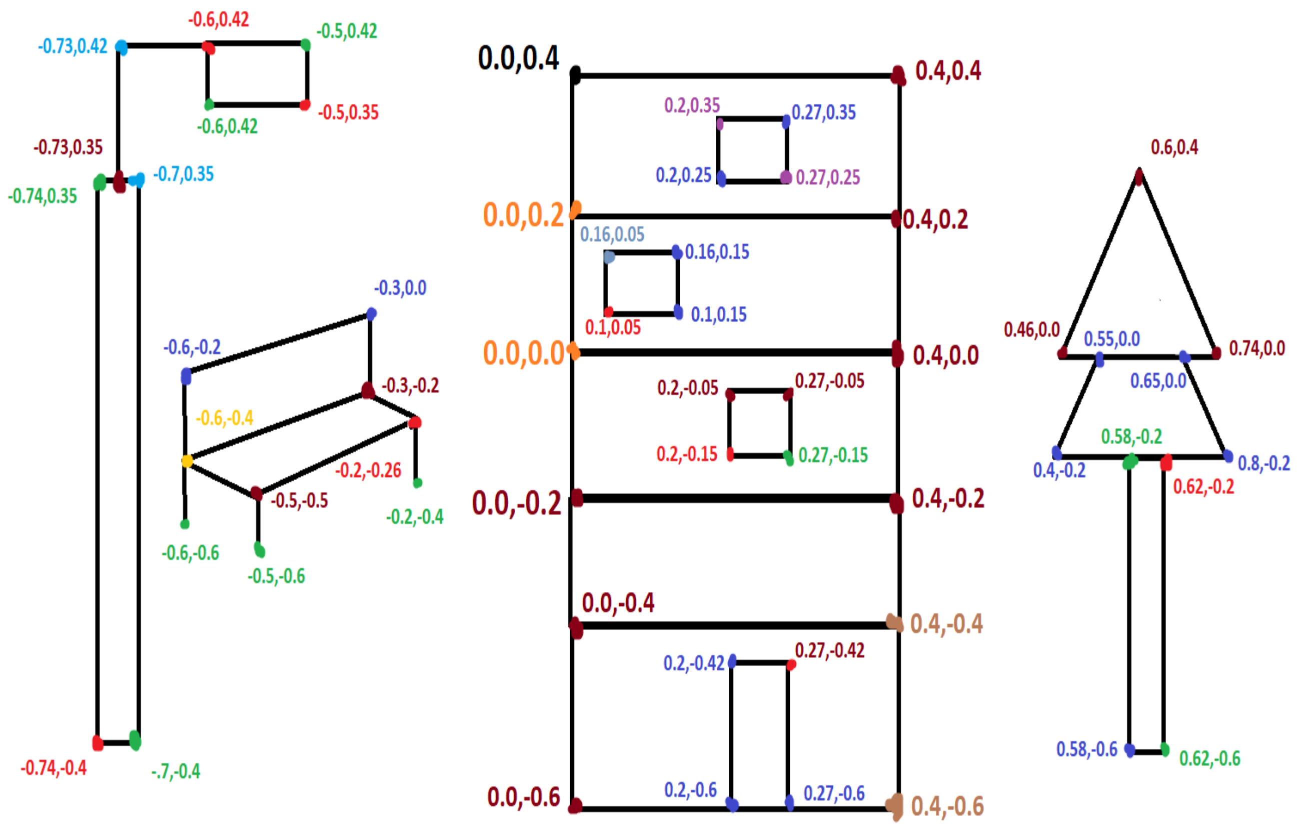
Output Screenshot (Full Screen)-



Question- 5

Use the building, tree, lamppost and bench to create a scenario

Graph Plot (Picture)-



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, 0.0f, 0.0f, 1.0f); // Black and opaque
}

/* 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
 glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(130, 80, 33  );
        glVertex2f(-0.74f, 0.35f);
        glVertex2f(-0.74f, -0.4f);
        glVertex2f(-0.7f, -0.4f);
glVertex2f(-0.7f, 0.35f);
glEnd();

glLineWidth(3);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(121, 125, 32 );
glVertex2f(-0.5f, 0.42f);
glVertex2f(-0.73f, 0.42f);
glVertex2f(-0.73f, 0.42f);
glVertex2f(-0.73f, 0.35f);
glEnd();

 glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(241, 252, 0 );
        glVertex2f(-0.5f, 0.42f);
        glVertex2f(-0.6f, 0.42f);
        glVertex2f(-0.6f, 0.35f);
glVertex2f(-0.5f, 0.35f);
glEnd();
 glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(121, 85, 57  );
        glVertex2f(-0.3f, 0.0f);
        glVertex2f(-0.6f, -0.2f);
        glVertex2f(-0.6f, -0.4f);
glVertex2f(-0.3f, -0.2f);
glEnd();

 glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(121, 85, 57  );
        glVertex2f(-0.3f, -0.2f);
        glVertex2f(-0.6f, -0.4f);
        glVertex2f(-0.5f, -0.5f);
glVertex2f(-0.2f, -0.26f);
glEnd();

glLineWidth(3);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(23, 32, 42 );
        glVertex2f(-0.3f, -0.2f);
        glVertex2f(-0.6f, -0.4f);
glEnd();


glLineWidth(9);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(23, 32, 42 );
        glVertex2f(-0.6f, -0.4f);
        glVertex2f(-0.6f, -0.6f);
glVertex2f(-0.2f, -0.26f);
        glVertex2f(-0.2f, -0.4f);
glVertex2f(-0.5f, -0.5f);
glVertex2f(-0.5f, -0.6f);

glEnd();


    glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(35, 103, 31  );
        glVertex2f(0.46f, 0.0f);
        glVertex2f(0.74f, 0.0f);
        glVertex2f(0.6f, 0.4f);
glEnd();


 glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(35, 103, 31  );
        glVertex2f(0.65f, 0.0f);
        glVertex2f(0.55f, 0.0f);
        glVertex2f(0.4f, -0.2f);
glVertex2f(0.8f, -0.2f);
glEnd();

glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(85, 62, 43   );
        glVertex2f(0.62f, -0.2f);
        glVertex2f(0.58f, -0.2f);
        glVertex2f(0.58f, -0.6f);
glVertex2f(0.62f, -0.6f);
glEnd();
```

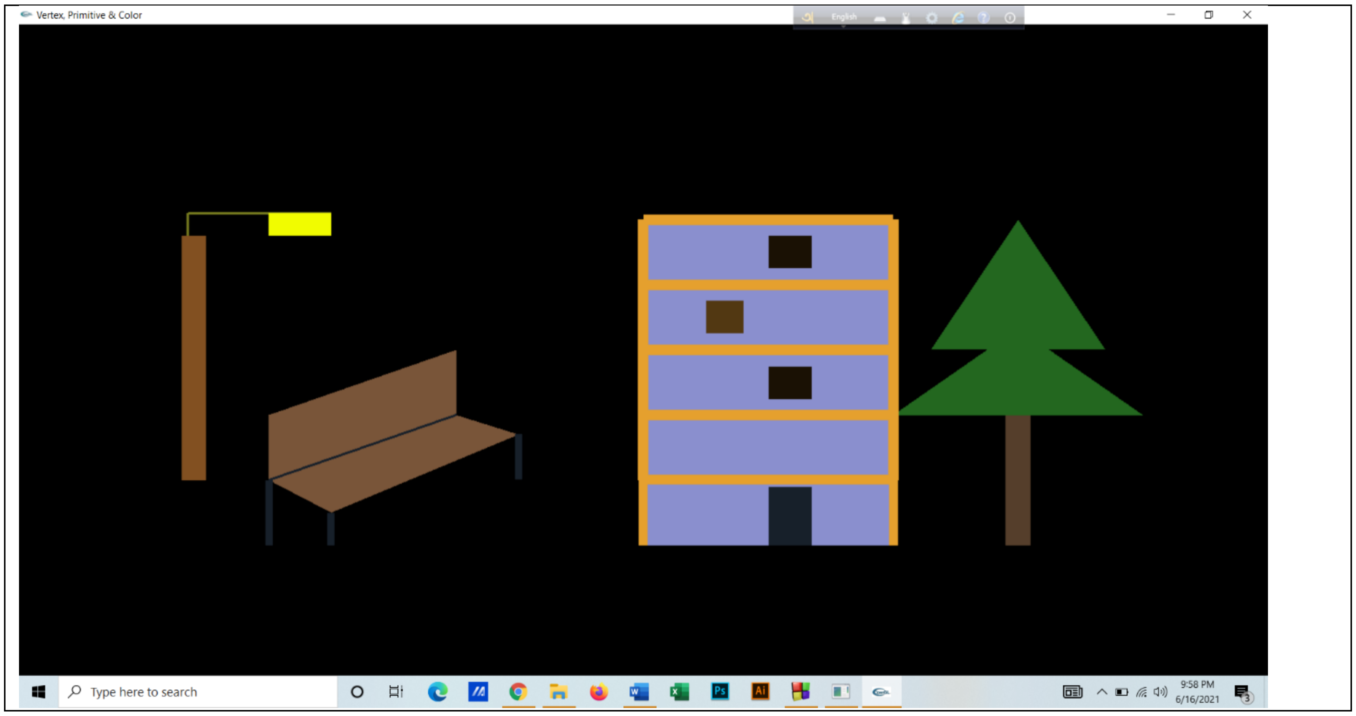
```
 glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(138, 143, 206  );
        glVertex2f(0.0f, 0.0f);
        glVertex2f(0.0f, -0.2f);
        glVertex2f(0.4f, -0.2f);
glVertex2f(0.4f, 0.0f);
glEnd();

glLineWidth(13);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(229, 160, 46  );
        glVertex2f(0.0f, 0.0f);
        glVertex2f(0.0f, -0.2f);
glVertex2f(0.0f, -0.2f);
glVertex2f(0.4f, -0.2f);
glVertex2f(0.4f, -0.2f);
        glVertex2f(0.4f, 0.0f);
        glVertex2f(0.4f, 0.0f);
        glVertex2f(0.0f, 0.0f);
glEnd();
 glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(138, 143, 206  );
        glVertex2f(0.0f, -0.2f);
        glVertex2f(0.0f, -0.4f);
        glVertex2f(0.4f, -0.4f);
glVertex2f(0.4f, -0.2f);
glEnd();

  glLineWidth(13);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(229, 160, 46 );
glVertex2f(0.0f, -0.2f);
        glVertex2f(0.0f, -0.4f);
glVertex2f(0.0f, -0.4f);
        glVertex2f(0.4f, -0.4f);
        glVertex2f(0.4f, -0.4f);
glVertex2f(0.4f, -0.2f);
glVertex2f(0.4f, -0.2f);
glVertex2f(0.0f, -0.2f);
glEnd();
```

```
 glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(138, 143, 206  );
        glVertex2f(0.0f, -0.4f);
        glVertex2f(0.0f, -0.6f);
        glVertex2f(0.4f, -0.6f);
glVertex2f(0.4f, -0.4f);
glEnd();

glLineWidth(11);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(229, 160, 46 );
        glVertex2f(0.0f, -0.4f);
        glVertex2f(0.0f, -0.6f);
        //glVertex2f(0.0f, -0.6f);
glVertex2f(0.4f, -0.6f);
//glVertex2f(0.4f, -0.6f);
glVertex2f(0.4f, -0.4f);
glVertex2f(0.4f, -0.4f);
        glVertex2f(0.0f, -0.4f);
 glEnd();

 glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(138, 143, 206  );
        glVertex2f(0.0f, 0.2f);
        glVertex2f(0.0f, 0.0f);
        glVertex2f(0.4f, 0.0f);
glVertex2f(0.4f, 0.2f);
glEnd();

glLineWidth(13);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(229, 160, 46 );
        glVertex2f(0.0f, 0.2f);
        glVertex2f(0.0f, 0.0f);
glVertex2f(0.0f, 0.0f);
        glVertex2f(0.4f, 0.0f);
glVertex2f(0.4f, 0.0f);
glVertex2f(0.4f, 0.2f);
glVertex2f(0.4f, 0.2f);
        glVertex2f(0.0f, 0.2f);
glEnd();
```

```
 glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(138, 143, 206  );
        glVertex2f(0.0f, 0.4f);
        glVertex2f(0.0f, 0.2f);
        glVertex2f(0.4f, 0.2f);
glVertex2f(0.4f, 0.4f);
glEnd();

 glLineWidth(13);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(229, 160, 46  );
        glVertex2f(0.0f, 0.4f);
        glVertex2f(0.0f, 0.2f);
glVertex2f(0.0f, 0.2f);
        glVertex2f(0.4f, 0.2f);
glVertex2f(0.4f, 0.2f);
glVertex2f(0.4f, 0.4f);
glVertex2f(0.4f, 0.4f);
glVertex2f(0.0f, 0.4f);
glEnd();

  glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(23, 32, 42 );
        glVertex2f(0.2f, -0.42f);
        glVertex2f(0.2f, -0.6f);
        glVertex2f(0.27f, -0.6f);
glVertex2f(0.27, -0.42f);
glEnd();

glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(26, 17, 4  );
        glVertex2f(0.2f, -0.15);
        glVertex2f(0.27f, -0.15f);
        glVertex2f(0.27f, -0.05f);
glVertex2f(0.2, -0.05f);
glEnd();


glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(82, 56, 18  );
        glVertex2f(0.1f, 0.05);
        glVertex2f(0.16f, 0.05f);
```

```
         glVertex2f(0.16f, 0.15f);
   glVertex2f(0.1f, 0.15f);
   glEnd();

   glBegin(GL_POLYGON);           // These vertices form a closed polygon
         glColor3ub(26, 17, 4  );
         glVertex2f(0.2f, 0.35);
         glVertex2f(0.2f, 0.25f);
         glVertex2f(0.27, 0.25f);
   glVertex2f(0.27f, 0.35f);
   glEnd();

         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)-


Vertex, Primitive & Color
Type here to search
9:58 PM
6/16/2021