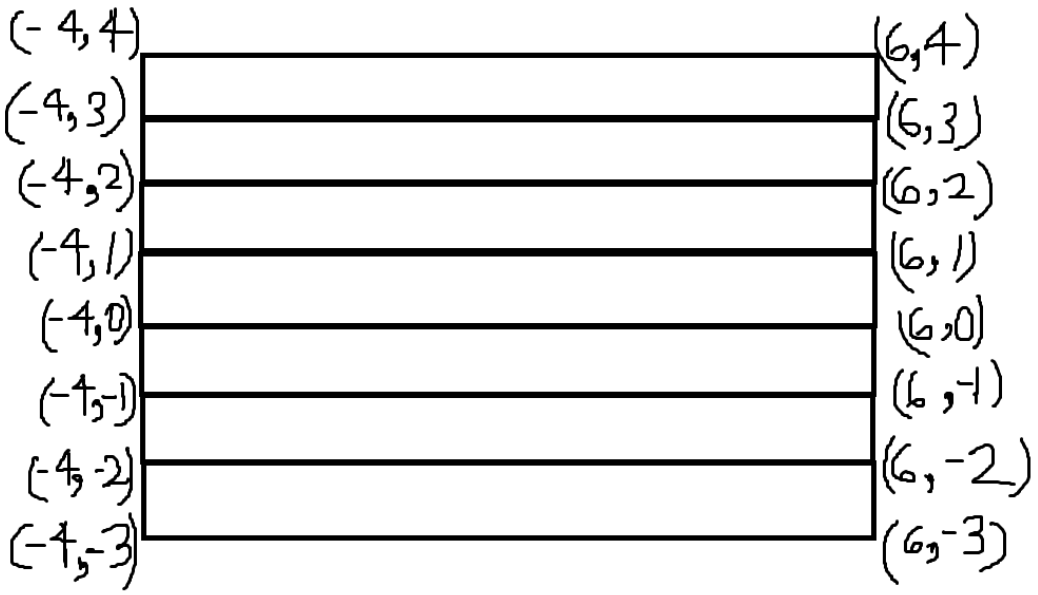
Lab Taks-2

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-2
- Must include resources for all the section in the table

Question- 1 Draw a Rainbow Flag 
Graph Plot (Picture)-



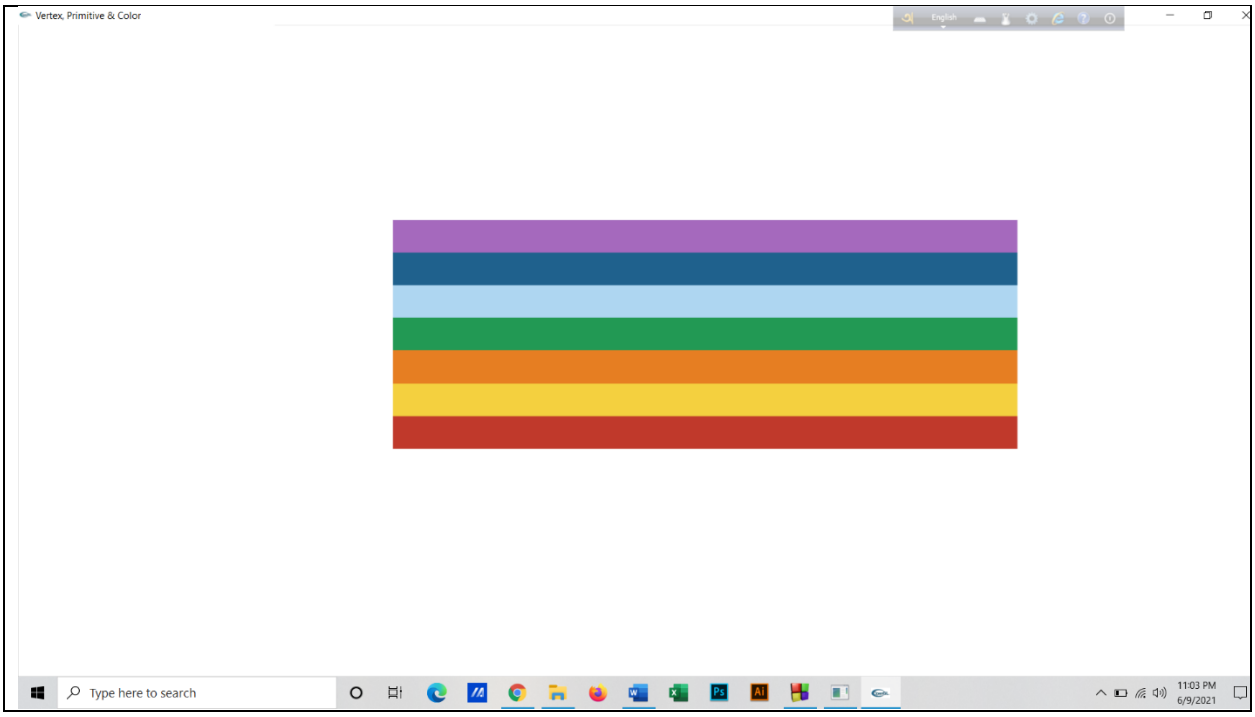
Code-

```
#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_QUADS);             // Each set of 4 vertices form a quad

  glColor3ub(165, 105, 189 );
        glVertex2f(-0.4f, +0.4f);    // x, y
```

```
        glVertex2f(+0.6f, +0.4f);
        glVertex2f(+0.6f, +0.3f);    // x, y
        glVertex2f(-0.4f, +0.3f);

        glEnd();

 glBegin(GL_QUADS);            // Each set of 4 vertices form a quad

glColor3ub(31, 97, 141 );
        glVertex2f(-0.4f, +0.3f);    // x, y
        glVertex2f(+0.6f, +0.3f);
        glVertex2f(+0.6f, +0.2f);    // x, y
        glVertex2f(-0.4f, +0.2f);

        glEnd();


 glBegin(GL_QUADS);            // Each set of 4 vertices form a quad

glColor3ub(174, 214, 241);
        glVertex2f(-0.4f, +0.2f);    // x, y
        glVertex2f(+0.6f, +0.2f);
        glVertex2f(+0.6f, +0.1f);    // x, y
        glVertex2f(-0.4f, +0.1f);

        glEnd();

        glBegin(GL_QUADS);            // Each set of 4 vertices form a quad

glColor3ub(34, 153, 84 );
        glVertex2f(-0.4f, +0.1f);    // x, y
        glVertex2f(+0.6f, +0.1f);
        glVertex2f(+0.6f, +0.0f);    // x, y
        glVertex2f(-0.4f, +0.0f);

        glEnd();
glBegin(GL_QUADS);            // Each set of 4 vertices form a quad

glColor3ub(230, 126, 34 );
        glVertex2f(-0.4f, +0.0f);    // x, y
        glVertex2f(+0.6f, +0.0f);
        glVertex2f(+0.6f, -0.1f);    // x, y
        glVertex2f(-0.4f, -0.1f);
```

```
        glEnd();


         glBegin(GL_QUADS);           // Each set of 4 vertices form a quad

  glColor3ub(244, 208, 63);
        glVertex2f(-0.4f, -0.1f);    // x, y
        glVertex2f(+0.6f, -0.1f);
        glVertex2f(+0.6f, -0.2f);    // x, y
        glVertex2f(-0.4f, -0.2f);

        glEnd();
         glBegin(GL_QUADS);           // Each set of 4 vertices form a quad

  glColor3ub(192, 57, 43);
        glVertex2f(-0.4f, -0.2f);    // x, y
        glVertex2f(+0.6f, -0.2f);
        glVertex2f(+0.6f, -0.3f);    // x, y
        glVertex2f(-0.4f, -0.3f);

        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;
}
```

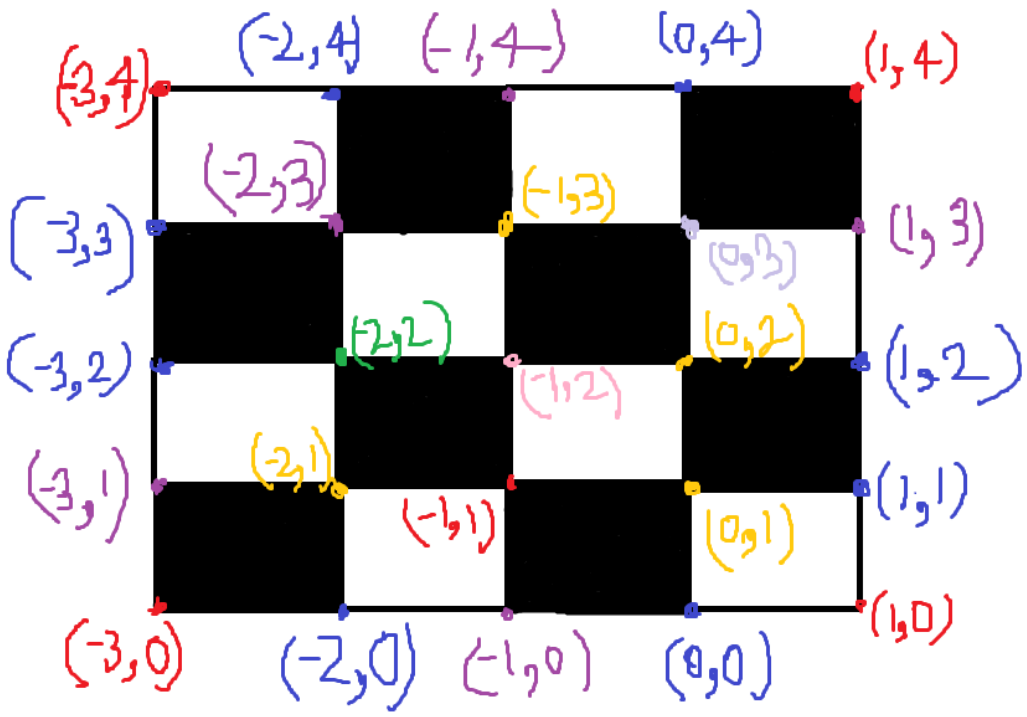
Output Screenshot (Full Screen)-



Question- 2

Draw 8X8 Chess Board

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, 1.0f, 1.0f, 0.0f); // 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_QUADS);             // Each set of 4 vertices form a quad

  glColor3ub(253, 254, 254 );
```

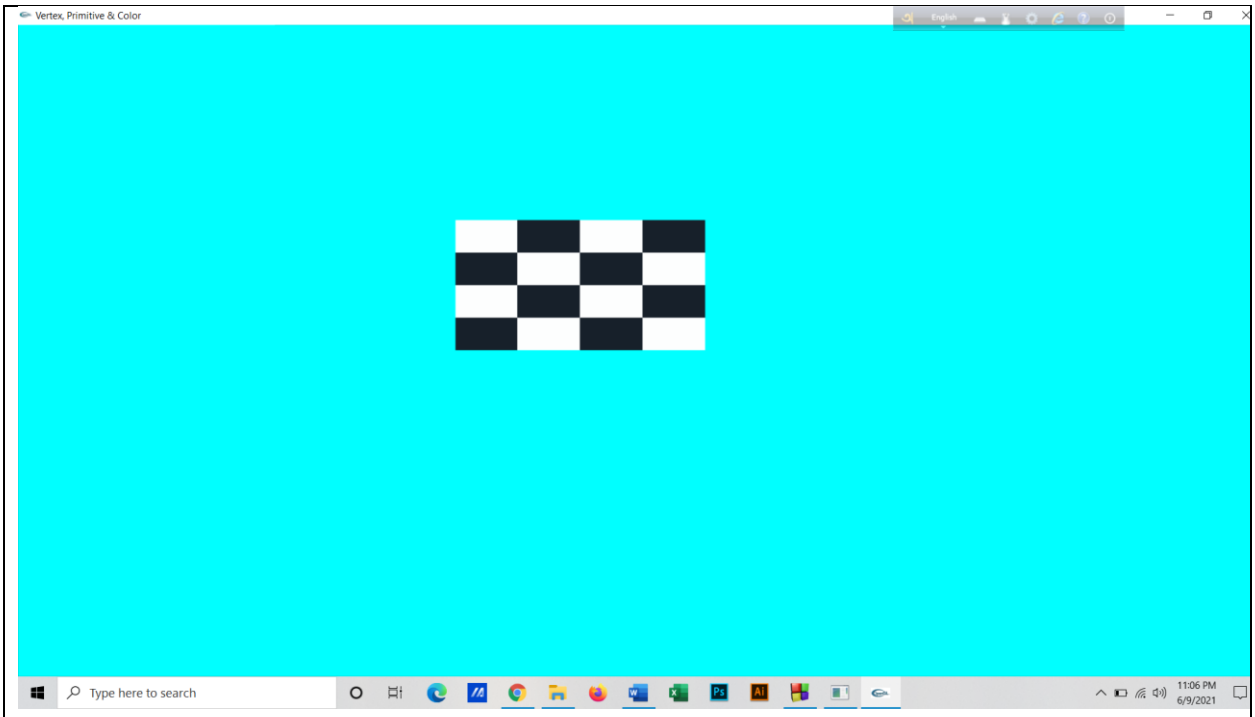
```
        glVertex2f(-0.3f, +0.4f);    // x, y
        glVertex2f(-0.2f, +0.4f);
        glVertex2f(-0.2f, +0.3f);    // x, y
        glVertex2f(-0.3f, +0.3f);

        glEnd();

 glBegin(GL_QUADS);           // Each set of 4 vertices form a quad

glColor3ub(23, 32, 42 );
        glVertex2f(-0.2f, +0.3f);    // x, y
        glVertex2f(-0.2f, +0.4f);
        glVertex2f(-0.1f, +0.4f);    // x, y
        glVertex2f(-0.1f, +0.3f);

        glEnd();


 glBegin(GL_QUADS);           // Each set of 4 vertices form a quad

glColor3ub(253, 254, 254);
        glVertex2f(-0.1f, +0.3f);    // x, y
        glVertex2f(-0.1f, +0.4f);
        glVertex2f(+0.0f, +0.4f);    // x, y
        glVertex2f(+0.0f, +0.3f);

        glEnd();

        glBegin(GL_QUADS);           // Each set of 4 vertices form a quad

glColor3ub(23, 32, 42 );
        glVertex2f(+0.0f, +0.3f);    // x, y
        glVertex2f(+0.0f, +0.4f);
        glVertex2f(+0.1f, +0.4f);    // x, y
        glVertex2f(+0.1f, +0.3f);

        glEnd();

glBegin(GL_QUADS);           // Each set of 4 vertices form a quad

glColor3ub(23, 32, 42);
        glVertex2f(-0.2f, +0.3f);    // x, y
        glVertex2f(-0.3f, +0.3f);
        glVertex2f(-0.3f, +0.2f);    // x, y
```

```
        glVertex2f(-0.2f, +0.2f);

        glEnd();


         glBegin(GL_QUADS);           // Each set of 4 vertices form a quad

glColor3ub(253, 254, 254);
        glVertex2f(-0.2f, 0.2f);    // x, y
        glVertex2f(-0.1f, 0.2f);
        glVertex2f(-0.1f, 0.3f);    // x, y
        glVertex2f(-0.2f, 0.3f);

        glEnd();

         glBegin(GL_QUADS);           // Each set of 4 vertices form a quad

glColor3ub(23, 32, 42);
        glVertex2f(-0.1f, 0.3f);    // x, y
        glVertex2f(-0.1f, 0.2f);
        glVertex2f(+0.0f, 0.2f);    // x, y
        glVertex2f(0.0f, 0.3f);

        glEnd();

         glBegin(GL_QUADS);           // Each set of 4 vertices form a quad

glColor3ub(253, 254, 254);
        glVertex2f(-0.3f, 0.1f);    // x, y
        glVertex2f(-0.2f, 0.1f);
        glVertex2f(-0.2f, 0.2f);    // x, y
        glVertex2f(-0.3f, 0.2f);

        glEnd();

        glBegin(GL_QUADS);           // Each set of 4 vertices form a quad

glColor3ub(253, 254, 254);
        glVertex2f(0.0f, 0.2f);    // x, y
        glVertex2f(0.1f, 0.2f);
        glVertex2f(+0.1f, 0.3f);    // x, y
        glVertex2f(0.0f, 0.3f);

        glEnd();
```

```
        glBegin(GL_QUADS);          // Each set of 4 vertices form a quad

    glColor3ub(23, 32, 42);
        glVertex2f(-0.2f, 0.1f);    // x, y
        glVertex2f(-0.1f, 0.1f);
        glVertex2f(-0.1f, 0.2f);    // x, y
        glVertex2f(-0.2f, 0.2f);

        glEnd();

        glBegin(GL_QUADS);          // Each set of 4 vertices form a quad

    glColor3ub(253, 254, 254);
        glVertex2f(-0.1f, 0.1f);    // x, y
        glVertex2f(0.0f, 0.1f);
        glVertex2f(0.0f, 0.2f);    // x, y
        glVertex2f(-0.1f, 0.2f);

        glEnd();

        glBegin(GL_QUADS);          // Each set of 4 vertices form a quad

    glColor3ub(23, 32, 42);
        glVertex2f(0.0f, 0.2f);    // x, y
        glVertex2f(0.0f, 0.1f);
        glVertex2f(0.1f, 0.1f);    // x, y
        glVertex2f(0.1f, 0.2f);

        glEnd();
        glBegin(GL_QUADS);          // Each set of 4 vertices form a quad

    glColor3ub(23, 32, 42);
        glVertex2f(-0.3f, 0.1f);    // x, y
        glVertex2f(-0.3f, 0.0f);
        glVertex2f(-0.2f, 0.0f);    // x, y
        glVertex2f(-0.2f, 0.1f);

        glEnd();
        glBegin(GL_QUADS);          // Each set of 4 vertices form a quad

    glColor3ub(253, 254, 254);
        glVertex2f(-0.2f, 0.1f);    // x, y
        glVertex2f(-0.2f, 0.0f);
        glVertex2f(-0.1f, 0.0f);    // x, y
```

```
        glVertex2f(-0.1f, 0.1f);

        glEnd();
        glBegin(GL_QUADS);          // Each set of 4 vertices form a quad

  glColor3ub(23, 32, 42);
        glVertex2f(-0.1f, 0.1f);   // x, y
        glVertex2f(-0.1f, 0.0f);
        glVertex2f(0.0f, 0.0f);    // x, y
        glVertex2f(0.0f, 0.1f);

        glEnd();
   glBegin(GL_QUADS);          // Each set of 4 vertices form a quad

  glColor3ub(253, 254, 254);
        glVertex2f(0.1f, 0.1f);    // x, y
        glVertex2f(0.0f, 0.1f);
        glVertex2f(0.0f, 0.0f);    // x, y
        glVertex2f(0.1f, 0.0f);

        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

Create the batman logo given below-

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(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
        glColor3f(1.0f, 1.0f, 0.0f); // Yellow
        glVertex2f(-0.4f, 0.0f);
        glVertex2f(-0.4f, -0.1f);
        glVertex2f(-0.2f, -0.4f);
        glVertex2f(0.0f, -0.4f);
        glVertex2f(0.2f, -0.4f);
        glVertex2f(0.4f, -0.1f);
   glVertex2f(0.4f, 0.0f);
   glVertex2f(0.4f, 0.1f);
   glVertex2f(0.2f, 0.4f);
   glVertex2f(0.0f, 0.4f);
   glVertex2f(-0.2f, 0.4f);
   glVertex2f(-0.4f, 0.1f);
   glEnd();
   glLineWidth(15);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(23, 32, 42 );
        glVertex2f(-0.4f, 0.0f);
        glVertex2f(-0.4f, -0.1f);
   glVertex2f(-0.4f, -0.1f);
        glVertex2f(-0.2f, -0.4f);
        glVertex2f(-0.2f, -0.4f);
        glVertex2f(0.0f, -0.4f);
   glVertex2f(0.0f, -0.4f);
        glVertex2f(0.2f, -0.4f);
        glVertex2f(0.2f, -0.4f);
        glVertex2f(0.4f, -0.1f);
   glVertex2f(0.4f, -0.1f);
   glVertex2f(0.4f, 0.0f);
   glVertex2f(0.4f, 0.0f);
   glVertex2f(0.4f, 0.1f);
   glVertex2f(0.4f, 0.1f);
   glVertex2f(0.2f, 0.4f);
   glVertex2f(0.2f, 0.4f);
   glVertex2f(0.0f, 0.4f);
```

```
glVertex2f(0.0f, 0.4f);
glVertex2f(-0.2f, 0.4f);
glVertex2f(-0.2f, 0.4f);
glVertex2f(-0.4f, 0.1f);
glVertex2f(-0.4f, 0.1f);
glVertex2f(-0.4f, 0.0f);


 glEnd();

 glBegin(GL_POLYGON);          // These vertices form a closed polygon
       glColor3ub(23, 32, 42 );
glVertex2f(0.0f, 0.3f);
       glVertex2f(0.0f, 0.0f);
       glVertex2f(0.2f, 0.0f);
glVertex2f(0.2f, 0.1f);
       glVertex2f(0.1f, 0.2f);
       glVertex2f(0.1f, 0.3f);


glEnd();


glBegin(GL_POLYGON);          // These vertices form a closed polygon
       glColor3ub(23, 32, 42 );
glVertex2f(0.2f, 0.1f);
       glVertex2f(0.2f, 0.0f);
       glVertex2f(0.3f, 0.0f);
glVertex2f(0.3f, 0.1f);
       glVertex2f(0.2f, 0.3f);
glEnd();


 glBegin(GL_POLYGON);          // These vertices form a closed polygon
       glColor3ub(23, 32, 42 );
glVertex2f(0.0f, 0.0f);
       glVertex2f(0.0f, -0.3f);
       glVertex2f(0.1f, -0.3f);
glVertex2f(0.1f, -0.2f);
       glVertex2f(0.2f, -0.1f);
       glVertex2f(0.2f, 0.0f);
glEnd();
```

```
 glBegin(GL_POLYGON);          // These vertices form a closed polygon
       glColor3ub(23, 32, 42 );
glVertex2f(0.2f, 0.0f);

       glVertex2f(0.2f, -0.3f);
       glVertex2f(0.3f, -0.1f);
glVertex2f(0.3f, 0.0f);




glEnd();

glBegin(GL_POLYGON);          // These vertices form a closed polygon
       glColor3ub(23, 32, 42 );
glVertex2f(0.0f, 0.3f);

       glVertex2f(-0.1f, 0.3f);
       glVertex2f(-0.1f, 0.2f);
glVertex2f(-0.2f, 0.1f);

       glVertex2f(-0.2f, 0.0f);
       glVertex2f(0.0f, 0.0f);
  glEnd();

 glBegin(GL_POLYGON);          // These vertices form a closed polygon
       glColor3ub(23, 32, 42 );
       glVertex2f(-0.2f, 0.0f);
       glVertex2f(-0.2f, 0.1f);
glVertex2f(-0.2f, 0.3f);

       glVertex2f(-0.3f, 0.1f);
glVertex2f(-0.3f, 0.0f);
glEnd();

 glBegin(GL_POLYGON);          // These vertices form a closed polygon
       glColor3ub(23, 32, 42 );
glVertex2f(0.0f,-0.3f);
       glVertex2f(0.0f, 0.0f);
       glVertex2f(-0.2f, 0.0f);
glVertex2f(-0.2f, -0.1f);
       glVertex2f(-0.1f,-0.2f);
glVertex2f(-0.1f,-0.3f);
```

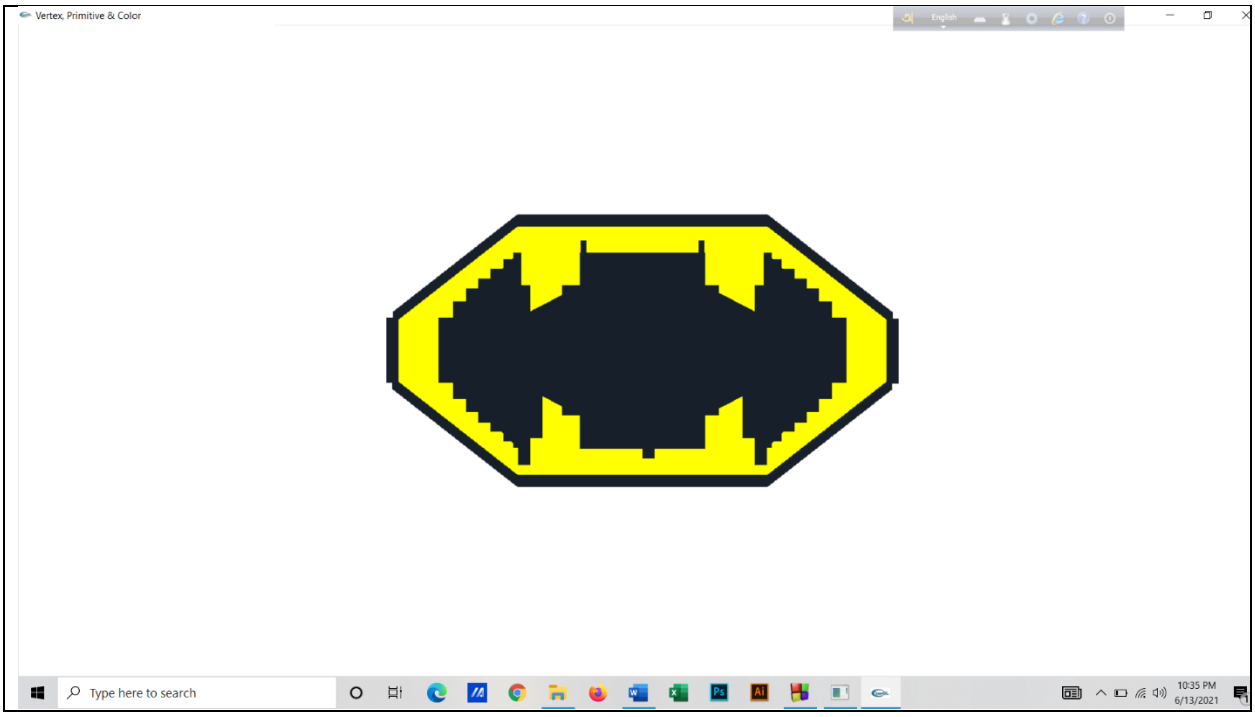
```
 glEnd();


glBegin(GL_POLYGON);          // These vertices form a closed polygon
       glColor3ub(23, 32, 42 );
       glVertex2f(-0.2f, 0.0f);
       glVertex2f(-0.3f, 0.0f);
glVertex2f(-0.3f, -0.1f);
       glVertex2f(-0.2f, -0.3f);

glEnd();

glLineWidth(10);
glBegin(GL_LINES);          // These vertices form a closed polygon
       glColor3ub(23, 32, 42 );
       glVertex2f(0.2f, 0.30f);
glVertex2f(0.2f,- 0.30f);
glEnd();
glLineWidth(30);
glBegin(GL_LINES);
 glVertex2f(0.09f,0.30f);
glVertex2f(0.1f,0.30f);

 glEnd();
 glLineWidth(30);
glBegin(GL_LINES);
 glVertex2f(-0.09f,0.30f);
glVertex2f(-0.1f,0.30f);
 glEnd();

glLineWidth(25);
glBegin(GL_LINES);
 glVertex2f(0.0f,-0.3f);
glVertex2f(0.02f,-0.3f);
 glEnd();
 glLineWidth(10);
glBegin(GL_LINES);          // These vertices form a closed polygon
       glColor3ub(23, 32, 42 );
       glVertex2f(-0.2f, 0.30f);
glVertex2f(-0.2f,- 0.30f);
glEnd();
glLineWidth(15);
glBegin(GL_LINES);
```

```
 glVertex2f(-0.19f,0.2f);
glVertex2f(-0.19f,-0.35f);
 glEnd();
 glLineWidth(15);
glBegin(GL_LINES);
 glVertex2f(0.19f,0.2f);
glVertex2f(0.19f,-0.35f);
 glEnd();
 glLineWidth(30);
glBegin(GL_LINES);
 glVertex2f(0.18f,0.1f);
glVertex2f(0.18f,-0.27f);
 glEnd();
 glLineWidth(30);
glBegin(GL_LINES);
 glVertex2f(-0.18f,0.1f);
glVertex2f(-0.18f,-0.27f);
 glEnd();
 glLineWidth(45);
glBegin(GL_LINES);
 glVertex2f(-0.1f,0.2f);
glVertex2f(-0.1f,-0.2f);
 glEnd();
 glLineWidth(35);
glBegin(GL_LINES);
 glVertex2f(0.1f,0.2f);
glVertex2f(0.1f,-0.2f);
 glEnd();
 glLineWidth(25);
glBegin(GL_LINES);
 glVertex2f(-0.3f,0.1f);
glVertex2f(-0.3f,-0.1f);
 glEnd();
 glLineWidth(25);
glBegin(GL_LINES);
 glVertex2f(0.3f,0.1f);
glVertex2f(0.3f,-0.1f);
 glEnd();
  glLineWidth(25);
glBegin(GL_LINES);
 glVertex2f(-0.31f,0.10f);
glVertex2f(-0.31f,-0.10f);
 glEnd();
  glLineWidth(25);
```

```
glBegin(GL_LINES);
 glVertex2f(0.31f,0.10f);
glVertex2f(0.31f,-0.10f);
 glEnd();
 glLineWidth(20);
glBegin(GL_LINES);
 glVertex2f(-0.29f,0.15f);
glVertex2f(-0.29f,-0.15f);
 glEnd();
 glLineWidth(20);
glBegin(GL_LINES);
 glVertex2f(0.29f,0.15f);
glVertex2f(0.29f,-0.15f);
 glEnd();
  glLineWidth(20);
glBegin(GL_LINES);
 glVertex2f(-0.27f,0.19f);
glVertex2f(-0.27f,-0.19f);
 glEnd();
 glLineWidth(20);
glBegin(GL_LINES);
 glVertex2f(0.27f,0.19f);
glVertex2f(0.27f,-0.19f);
 glEnd();
  glLineWidth(20);
glBegin(GL_LINES);
 glVertex2f(-0.25f,0.22f);
glVertex2f(-0.25f,-0.22f);
 glEnd();
 glLineWidth(20);
glBegin(GL_LINES);
 glVertex2f(0.25f,0.22f);
glVertex2f(0.25f,-0.22f);
 glEnd();
  glLineWidth(20);
glBegin(GL_LINES);
 glVertex2f(-0.23f,0.25f);
glVertex2f(-0.23f,-0.25f);
 glEnd();
  glLineWidth(20);
glBegin(GL_LINES);
 glVertex2f(0.23f,0.25f);
glVertex2f(0.23f,-0.25f);
 glEnd();
```

```
     glLineWidth(20);
   glBegin(GL_LINES);
    glVertex2f(-0.21f,0.28f);
   glVertex2f(-0.21f,-0.28f);
    glEnd();
     glLineWidth(20);
   glBegin(GL_LINES);
    glVertex2f(0.21f,0.28f);
   glVertex2f(0.21f,-0.28f);
    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
10:35 PM
6/13/2021