**Lab Taks-2**

* Rename the file with your serial number only.
* 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-  /\*  \* GL02Primitive.cpp: Vertex, Primitive and Color  \* Draw Simple 2D colored Shapes: quad, triangle and polygon.  \*/  #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);  glLineWidth(0.1f); // Set border thickness  glColor3f(0.0f, 0.0f, 0.0f); // Set border color to white  glBegin(GL\_LINE\_LOOP); // Draw the border  glVertex2f(-0.7f, 0.5f); // A (Top-left)  glVertex2f(-0.7f, -0.2f); // Q (Bottom-left)  glVertex2f(0.7f, -0.2f); // R (Bottom-right)  glVertex2f(0.7f, 0.5f); // D (Top-right)  glEnd();  // Draw filled quadrilateral with purple color  glColor3f(0.49f, 0.23f, 0.59f); // Purple  glBegin(GL\_QUADS);  glVertex2f(-0.7f, 0.5f); // A (Top-left)  glVertex2f(-0.7f, 0.4f); // E (Bottom-left)  glVertex2f(0.7f, 0.4f); // F (Bottom-right)  glVertex2f(0.7f, 0.5f); // D (Top-right)  glEnd();  // Draw black border  glLineWidth(0.1f); // Set border thickness  glColor3f(0.0f, 0.0f, 0.0f); // Black border  glBegin(GL\_LINE\_LOOP);  glVertex2f(-0.7f, 0.5f); // A (Top-left)  glVertex2f(-0.7f, 0.4f); // E (Bottom-left)  glVertex2f(0.7f, 0.4f); // F (Bottom-right)  glVertex2f(0.7f, 0.5f); // D (Top-right)  glEnd();  glBegin(GL\_QUADS); // Start drawing a quadrilateral (square)  glColor3f(0.13f, 0.38f, 0.54f);  glVertex2f(-0.7f,0.4f);//E // Bottom-left  glVertex2f(0.7f,0.4f);//F // Bottom-right  glVertex2f(0.7f,0.3f);//H // Top-right  glVertex2f(-0.7f,0.3f);//G  glEnd();  glLineWidth(0.1f); // Set border thickness  glColor3f(0.0f, 0.0f, 0.0f); // Black border  glBegin(GL\_LINE\_LOOP);  glVertex2f(-0.7f,0.4f);//E // Bottom-left  glVertex2f(0.7f,0.4f);//F // Bottom-right  glVertex2f(0.7f,0.3f);//H // Top-right  glVertex2f(-0.7f,0.3f);//G  glEnd();  glBegin(GL\_QUADS); // Start drawing a quadrilateral (square)  glColor3f(0.28f, 0.78f, 0.69f);  glVertex2f(-0.7f,0.3f);//G // Bottom-left  glVertex2f(0.7f,0.3f);//H // Bottom-right  glVertex2f(0.7f,0.2f);//J // Top-right  glVertex2f(-0.7f,0.2f);//I  glEnd();  glLineWidth(0.1f); // Set border thickness  glColor3f(0.0f, 0.0f, 0.0f); // Black border  glBegin(GL\_LINE\_LOOP);  glVertex2f(-0.7f,0.3f);//G // Bottom-left  glVertex2f(0.7f,0.3f);//H // Bottom-right  glVertex2f(0.7f,0.2f);//J // Top-right  glVertex2f(-0.7f,0.2f);//I  glEnd();  glBegin(GL\_QUADS); // Start drawing a quadrilateral (square)  glColor3f(0.13f, 0.6f, 0.329f);  glVertex2f(-0.7f,0.2f);//I // Bottom-left  glVertex2f(0.7f,0.2f);//J // Bottom-right  glVertex2f(0.7f,0.1f);//L // Top-right  glVertex2f(-0.7f,0.1f);//K  glEnd();  glLineWidth(0.1f); // Set border thickness  glColor3f(0.0f, 0.0f, 0.0f); // Black border  glBegin(GL\_LINE\_LOOP);  glVertex2f(-0.7f,0.2f);//I // Bottom-left  glVertex2f(0.7f,0.2f);//J // Bottom-right  glVertex2f(0.7f,0.1f);//L // Top-right  glVertex2f(-0.7f,0.1f);//K  glEnd();  glBegin(GL\_QUADS); // Start drawing a quadrilateral (square)  glColor3f(0.95f, 0.61f, 0.07f);  glVertex2f(-0.7f,0.1f);//K // Bottom-left  glVertex2f(0.7f,0.1f);//L // Bottom-right  glVertex2f(0.7f,0.0f);//N // Top-right  glVertex2f(-0.7f,0.0f);//M  glEnd();  glLineWidth(0.1f); // Set border thickness  glColor3f(0.0f, 0.0f, 0.0f); // Black border  glBegin(GL\_LINE\_LOOP);  glVertex2f(-0.7f,0.1f);//K // Bottom-left  glVertex2f(0.7f,0.1f);//L // Bottom-right  glVertex2f(0.7f,0.0f);//N // Top-right  glVertex2f(-0.7f,0.0f);//M  glEnd();  glBegin(GL\_QUADS); // Start drawing a quadrilateral (square)  glColor3f(0.96f, 0.861f, 0.43f);  glVertex2f(-0.7f,0.0f);//M // Bottom-left  glVertex2f(0.7f,0.0f);//N // Bottom-right  glVertex2f(0.7f,-0.1f);//O // Top-right  glVertex2f(-0.7f,-0.1f);//P  glEnd();  glLineWidth(0.1f); // Set border thickness  glColor3f(0.0f, 0.0f, 0.0f); // Black border  glBegin(GL\_LINE\_LOOP);  glVertex2f(-0.7f,0.0f);//M // Bottom-left  glVertex2f(0.7f,0.0f);//N // Bottom-right  glVertex2f(0.7f,-0.1f);//O // Top-right  glVertex2f(-0.7f,-0.1f);//P  glEnd();  glBegin(GL\_QUADS); // Start drawing a quadrilateral (square)  glColor3f(0.91f, 0.29f, 0.23f);  glVertex2f(0.7f,-0.1f);//O // Bottom-left  glVertex2f(-0.7f,-0.1f);//P // Bottom-right  glVertex2f(-0.7f,-0.2f);//Q // Top-right  glVertex2f(0.7f,-0.2f);//R  glEnd();  glLineWidth(0.1f); // Set border thickness  glColor3f(0.0f, 0.0f, 0.0f); // Black border  glBegin(GL\_LINE\_LOOP);  glVertex2f(0.7f,-0.1f);//O // Bottom-left  glVertex2f(-0.7f,-0.1f);//P // Bottom-right  glVertex2f(-0.7f,-0.2f);//Q // Top-right  glVertex2f(0.7f,-0.2f);//R  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  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner  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)-**  **A computer screen with a black screen and a black screen  AI-generated content may be incorrect.** |

|  |
| --- |
| **Question- 2**  Flag of 5 different countries as per your choice. |
| **Graph Plot (Picture)-**    1.Bangladesh 2.Germany    3.Denmark 4. Finland    5. Palestine |
| **Code-**  **1.Bangladesh:**  #include <windows.h> // for MS Windows  #include <GL/glut.h> // GLUT, include glu.h and gl.h  #include <math.h>  /\* Handler for window-repaint event. Call back when the window first appears and  whenever the window needs to be re-painted. \*/  void display() {  glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer (background)  glLineWidth(7.5);  glBegin(GL\_POLYGON); // These vertices form a closed polygon  glColor3f(0.09f,0.43f,0.24f); // Green  glVertex2f(-0.6f, 0.4f);  glVertex2f(-0.6f, -0.4f);  glVertex2f(0.6f, -0.4f);  glVertex2f(0.6f, 0.4f);  glEnd();  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3f(1.0,0,0.0);//Red  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=0.2;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x,y );  }  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("OpenGL Setup Test");  //gluOrtho2D(-0.1,0.7,-0.1,0.3); // Create a window with the given title  glutInitWindowSize(320, 320);// Set the window's initial width & height  glutDisplayFunc(display);// Register display callback handler for window re-paint  glutMainLoop(); // Enter the event-processing loop  return 0;  }  **2. Germany:**  /\*  \* GL02Primitive.cpp: Vertex, Primitive and Color  \* Draw Simple 2D colored Shapes: quad, triangle and polygon.  \*/  #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);  }  /\* 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(0.09f, 0.12f, 0.16f);  glVertex2f(-0.32f, 0.2f);  glVertex2f(0.28f, 0.2f);  glVertex2f(0.28f, 0.08f);  glVertex2f(-0.32f, 0.08f);  glEnd();  glBegin(GL\_POLYGON); // These vertices form a closed polygon  glColor3f(0.91f, 0.29f, 0.24f);  glVertex2f(-0.32f, 0.08f);  glVertex2f(0.28f, 0.08f);  glVertex2f(0.28f, -0.04f);  glVertex2f(-0.32f, -0.04f);  glEnd();  glBegin(GL\_POLYGON); // These vertices form a closed polygon  glColor3f(0.95f, 0.82f, 0.24f);  glVertex2f(-0.32f, -0.04f);  glVertex2f(0.28f, -0.04f);  glVertex2f(0.28f, -0.16f);  glVertex2f(-0.32f, -0.16f);  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  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner  glutDisplayFunc(display); // Register callback handler for window re-paint event  initGL(); // Our own OpenGL initialization  glutMainLoop(); // Enter the event-processing loop  return 0;  }  **3. Denmark:**  /\*  \* GL02Primitive.cpp: Vertex, Primitive and Color  \* Draw Simple 2D colored Shapes: quad, triangle and polygon.  \*/  #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);  }  /\* 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(0.91f, 0.29f, 0.24f);  glVertex2f(-0.4f, 0.04f);//A  glVertex2f(-0.4f, 0.2f);//B  glVertex2f(-0.2f, 0.2f);//C  glVertex2f(-0.2f, 0.04f);//D  glEnd();  glBegin(GL\_POLYGON); // These vertices form a closed polygon  glColor3f(0.91f, 0.29f, 0.24f);  glVertex2f(-0.4f, -0.04f);//E  glVertex2f(-0.2f, -0.04f);//F  glVertex2f(-0.2f, -0.2f);//H  glVertex2f(-0.4f, -0.2f);//G  glEnd();  glBegin(GL\_POLYGON); // These vertices form a closed polygon  glColor3f(0.91f, 0.29f, 0.24f);  glVertex2f(-0.12f, 0.2f);//I  glVertex2f(0.24f, 0.2f);//J  glVertex2f(0.24f, 0.04f);//L  glVertex2f(-0.12f, 0.04f);//K  glEnd();  glBegin(GL\_POLYGON); // These vertices form a closed polygon  glColor3f(0.91f, 0.29f, 0.24f);  glVertex2f(-0.12f, -0.04f);//M  glVertex2f(0.24f, -0.04f);//N  glVertex2f(0.24f, -0.2f);//P  glVertex2f(-0.12f, -0.2f);//O  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  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner  glutDisplayFunc(display); // Register callback handler for window re-paint event  initGL(); // Our own OpenGL initialization  glutMainLoop(); // Enter the event-processing loop  return 0;  }  **4. Finland:**  /\*  \* GL02Primitive.cpp: Vertex, Primitive and Color  \* Draw Simple 2D colored Shapes: quad, triangle and polygon.  \*/  #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);  }  /\* 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,1.0f);  glVertex2f(-0.8f, 0.6f);  glVertex2f(-0.8f, -0.6f);  glVertex2f(0.6f, -0.6f);  glVertex2f(0.6f, 0.6f);  glEnd();  glColor3f(0.0f, 0.0f, 0.0f); // Black color for the border  glBegin(GL\_LINE\_LOOP); // This draws the border as a closed loop of lines  glVertex2f(-0.8f, 0.6f);  glVertex2f(-0.8f, -0.6f);  glVertex2f(0.6f, -0.6f);  glVertex2f(0.6f, 0.6f);  glEnd();  glBegin(GL\_POLYGON); // These vertices form a closed polygon  glColor3f(0.12f, 0.38f, 0.54f);  glVertex2f(-0.8f, 0.08f);  glVertex2f(-0.8f, -0.08f);  glVertex2f(0.6f, -0.08f);  glVertex2f(0.6f, 0.08f);  glEnd();  glBegin(GL\_POLYGON); // These vertices form a closed polygon  glColor3f(0.12f, 0.38f, 0.54f);  glVertex2f(-0.28f, 0.6f);  glVertex2f(-0.12, 0.6f);  glVertex2f(-0.12f, -0.6f);  glVertex2f(-0.28f, -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  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner  glutDisplayFunc(display); // Register callback handler for window re-paint event  initGL(); // Our own OpenGL initialization  glutMainLoop(); // Enter the event-processing loop  return 0;  }  **5. Palestine:**  /\*  \* GL02Primitive.cpp: Vertex, Primitive and Color  \* Draw Simple 2D colored Shapes: quad, triangle and polygon.  \*/  #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);  }  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(0.0f, 0.0f, 0.0f);  glVertex2f(-0.6f, 0.4f);  glVertex2f(0.6f, 0.4f);  glVertex2f(0.6f, 0.2f);  glVertex2f(-0.6f, 0.2f);  glEnd();  glBegin(GL\_POLYGON); // These vertices form a closed polygon  glColor3f(1.0f, 1.0f, 1.0f);  glVertex2f(-0.6f, 0.2f);  glVertex2f(0.6f, 0.2f);  glVertex2f(0.6f, 0.0f);  glVertex2f(-0.6f, 0.0f);  glEnd();  glBegin(GL\_POLYGON); // These vertices form a closed polygon  glColor3f(0.11f, 0.51f, 0.28f);  glVertex2f(-0.6f, 0.0f);  glVertex2f(0.6f, 0.0f);  glVertex2f(0.6f, -0.2f);  glVertex2f(-0.6f, -0.2f);  glEnd();  glBegin(GL\_TRIANGLES); // These vertices form a closed polygon  glColor3f(0.75f, 0.22f, 0.16f);  glVertex2f(-0.6f, 0.4f);  glVertex2f(-0.6f, -0.2f);  glVertex2f(-0.25f, 0.12f);  glEnd();  glLineWidth(1.0f); // Set the border thickness  glColor3f(0.0f, 0.0f, 0.0f); // Black color  glBegin(GL\_LINE\_LOOP);  glVertex2f(0.6f, 0.4f);  glVertex2f(0.6f, -0.2f);  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  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner  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)-**  **A computer screen with a black screen  AI-generated content may be incorrect.**  **A computer screen with a black screen  AI-generated content may be incorrect. A computer screen with a black screen  AI-generated content may be incorrect.**    **A computer screen with a black screen  AI-generated content may be incorrect.** |

|  |
| --- |
| **Question- 2**  Draw 4X4 Chess Board |
| **Graph Plot (Picture)-** |
| **Code-**  /\*  \* GL02Primitive.cpp: Vertex, Primitive and Color  \* Draw Simple 2D colored Shapes: quad, triangle and polygon.  \*/  #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  // Draw the black border first  glLineWidth(2.0f); // Set border thickness  glColor3f(0.0f, 0.0f, 0.0f); // Set border color to black  glBegin(GL\_LINE\_LOOP);  glVertex2f(-0.8f, 0.6f); // Bottom-left  glVertex2f(0.8f, 0.6f); // Bottom-right  glVertex2f(0.8f, -0.64f); // Top-right  glVertex2f(-0.8f, -0.64f); // Top-left  glEnd();  // Draw the white-filled quadrilateral  glColor3f(1.0f, 1.0f, 1.0f); // White fill  glBegin(GL\_QUADS);  glVertex2f(-0.8f, 0.6f);  glVertex2f(0.8f, 0.6f);  glVertex2f(0.8f, -0.644f);  glVertex2f(-0.8f, -0.6f);  glEnd();  glBegin(GL\_QUADS);  glColor3f(0.0f,0.0f,0.0f);  glVertex2f(-0.8f,0.6f);//A // Bottom-left  glVertex2f(-0.4f,0.6f);//E // Bottom-right  glVertex2f(-0.4f,0.28f);//F // Top-right  glVertex2f(-0.8f,0.28f);//G  glEnd();  glBegin(GL\_QUADS);  glColor3f(0.0f,0.0f,0.0f);  glVertex2f(0.0f,0.6f);//H // Bottom-left  glVertex2f(0.0f,0.28f);//I // Bottom-right  glVertex2f(0.4f,0.28f);//K // Top-right  glVertex2f(0.4f,0.6f);//J  glEnd();  glBegin(GL\_QUADS);  glColor3f(0.0f,0.0f,0.0f);  glVertex2f(-0.4f,0.28f);//F // Bottom-left  glVertex2f(0.0f,0.28f);//I // Bottom-right  glVertex2f(0.0f,-0.04f);//M // Top-right  glVertex2f(-0.4f,-0.04f);//L  glEnd();  glBegin(GL\_QUADS);  glColor3f(0.0f,0.0f,0.0f);  glVertex2f(0.4f,0.29f);//F // Bottom-left  glVertex2f(0.8f,0.29f);//I // Bottom-right  glVertex2f(0.8f,-0.04f);//M // Top-right  glVertex2f(0.4f,-0.04f);//L  glEnd();  glBegin(GL\_QUADS);  glColor3f(0.0f,0.0f,0.0f);  glVertex2f(-0.8f,-0.004f);//Q // Bottom-left  glVertex2f(-0.4f,-0.004f);//L // Bottom-right  glVertex2f(-0.4f,-0.32f);//S // Top-right  glVertex2f(-0.8f,-0.32f);//R  glEnd();  glBegin(GL\_QUADS);  glColor3f(0.0f,0.0f,0.0f);  glVertex2f(-0.4f,-0.32f);//S // Bottom-left  glVertex2f(0.0f,-0.32f);//U // Bottom-right  glVertex2f(0.0f,-0.64f);//V // Top-right  glVertex2f(-0.4f,-0.64f);//T  glEnd();  glBegin(GL\_QUADS);  glColor3f(0.0f,0.0f,0.0f);  glVertex2f(0.0f,-0.04f);//M // Bottom-left  glVertex2f(0.4f,-0.04f);//O // Bottom-right  glVertex2f(0.4f,-0.32f);//W // Top-right  glVertex2f(0.0f,-0.32f);//U  glEnd();  glBegin(GL\_QUADS);  glColor3f(0.0f,0.0f,0.0f);  glVertex2f(0.4f,-0.32f);//W // Bottom-left  glVertex2f(0.8f,-0.32f);//A1 // Bottom-right  glVertex2f(0.8f,-0.64f);//D // Top-right  glVertex2f(0.4f,-0.64f);//Z  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  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner  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)-**  **A computer screen shot of a black and white checkered background  AI-generated content may be incorrect.** |

|  |
| --- |
| **Question- 3**  Implement the given picture |
| **Graph Plot (Picture)-**  A screenshot of a graph  AI-generated content may be incorrect. |
| **Code-**  /\*  \* GL02Primitive.cpp: Vertex, Primitive and Color  \* Draw Simple 2D colored Shapes: quad, triangle and polygon.  \*/  #include <windows.h>  #include <GL/glut.h>  #include <math.h>  /\* Initialize OpenGL Graphics \*/  void initGL() {  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);  }  /\* 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);  glLineWidth(2.0f);  glColor3ub(0, 0, 0);  glBegin(GL\_LINES);  glVertex2f(-35, 5); // A  glVertex2f(-25, 25); // B  glEnd();  glBegin(GL\_LINES);  glVertex2f(-25, 25);  glVertex2f(-20, 10);  glEnd();  glBegin(GL\_LINES);  glVertex2f(-20, 10);  glVertex2f(-10, 20);  glEnd();  glBegin(GL\_LINES);  glVertex2f(-10, 20);  glVertex2f(-5, 15);  glEnd();  glBegin(GL\_LINES);  glVertex2f(-5, 15);  glVertex2f(5, 10);  glEnd();  glBegin(GL\_LINES);  glVertex2f(5, 10);  glVertex2f(10, 5);  glEnd();  glBegin(GL\_LINES);  glVertex2f(-30, 0);//h  glVertex2f(10, 0);//l  glEnd();  glBegin(GL\_LINES);  glVertex2f(-10, 5);  glVertex2f(-10, 0);//l  glEnd();  glBegin(GL\_LINES);  glVertex2f(-13.21f,5.0f);  glVertex2f(-10, 5);//l  glEnd();  //glBegin(GL\_LINES);  //glVertex2f(-10.0f,5.0f);  //glVertex2f(-7.20f,5.18f);//l  //glEnd();  // glBegin(GL\_LINES);  // glVertex2f(-7.20f,5.18f);//  // glVertex2f(-10.0f,9.0f);  //glEnd();  // glBegin(GL\_LINES);  // glVertex2f(-10.0f,9.0f);  // glVertex2f(-13.21f,5.0f);  //glEnd();  // glBegin(GL\_LINES);  // glVertex2f(-10.0f,9.0f);  //glVertex2f(-13.60f,8.0f);  //glEnd();  glBegin(GL\_POLYGON);  glColor3ub(0, 0, 0); //Black  glVertex2f(-10.0f, 5.0f);  glVertex2f(-7.20f, 5.18f);  glVertex2f(-10.0f, 9.0f);  glVertex2f(-13.21f, 5.0f);  glVertex2f(-10.60f, 5.0f);  glEnd();  // glBegin(GL\_LINES);  //glVertex2f(-13.60f,8.0f);  // glVertex2f(-10.0f,12.50f);  //glEnd();  //glBegin(GL\_POLYGON);  // glColor3ub(0,0,0);  //glBegin(GL\_LINES);  // glVertex2f(-13.60f,8.0f);  // glVertex2f(-10.0f,12.50f);  //glEnd();  //glBegin(GL\_POLYGON);  //glColor3ub(0,0,0);  //glBegin(GL\_LINES);  //glVertex2f(-10.0f,12.0f);  //glVertex2f(-6.20f,8.550f);  //glEnd();  //glBegin(GL\_POLYGON);  //glColor3ub(0,0,0);  //glVertex2f(-6.02f,8.55f);  //glVertex2f(-10.0f,9.02f);  //glEnd();  glBegin(GL\_POLYGON);  glColor3ub(0, 0, 0);  glVertex2f(-13.60f, 8.0f);  glVertex2f(-10.0f, 12.50f);  glVertex2f(-6.20f, 8.55f);  glEnd();  float cx = -28.0f; // B point er X  float cy = 30.0f; // B point er upor  float r = 3.19f; // GeoGebra radius  int segments = 100;  glColor3ub(255, 204, 0);//yellow sun  glBegin(GL\_POLYGON);  for (int i = 0; i < segments; ++i) {  float theta = 2.0f \* 3.1415926f \* i / segments;  float x = r \* cosf(theta);  float y = r \* sinf(theta);  glVertex2f(cx + x, cy + y);  }  glEnd();  glFlush();  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitWindowSize(800, 600); // Set window size  glutInitWindowPosition(50, 50); // Optional: set window position  glutCreateWindow("Custom Shape with Circle");  gluOrtho2D(-40, 40, -10, 40); // Our own OpenGL initialization  initGL();  glutDisplayFunc(display);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |