Lab Practice-7

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.

Question-
Create a simple day and night scenario that will automatically change from day to night

Code-

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

GLfloat position = 0.0f;
GLfloat position1 = 0.0f;

GLfloat speed = 0.1f;
void dis();
void display();

void update(int value) {

   if(position <-1.5)
      position = 1.0f;

   position -= speed;

         glutPostRedisplay();


         glutTimerFunc(20,update,0);
}


void update1(int value) {

   if(position1 >1.0)
      position1 = -1.0f;
   position1 += speed;

          glutPostRedisplay();


          glutTimerFunc(20,update1,0);
}

/* 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 disback(int val)
{
   glutDisplayFunc(display);
}






void display3()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
glPushMatrix();

glBegin(GL_POLYGON);           // These vertices form a closed polygon
          glColor3ub(18, 62, 19 );
          glVertex2f(-0.99f, -0.2f);
          glVertex2f(0.99f, 0.0f);
          glVertex2f(0.99f, -0.99f);
   glVertex2f(-0.99f, -0.99f);
   glEnd();
    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(0.36f, 0.4f);
        glVertex2f(0.64f, 0.4f);
        glVertex2f(0.5f, 0.8f);
glEnd();


 glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(35, 103, 31  );
        glVertex2f(0.55f, 0.4f);
        glVertex2f(0.45f, 0.4f);
        glVertex2f(0.3f, 0.2f);
glVertex2f(0.7f, 0.2f);
glEnd();

glBegin(GL_POLYGON);          // These vertices form a closed polygon
        glColor3ub(85, 62, 43   );
        glVertex2f(0.52f, -0.2f);
        glVertex2f(0.48f, -0.2f);
        glVertex2f(0.48f, 0.2f);
glVertex2f(0.52f, 0.2f);
glEnd();


    glBegin(GL_POLYGON);          // These vertices form a closed polygon
```

```
            glColor3ub(35, 103, 31  );
            glVertex2f(0.0f, 0.2f);
            glVertex2f(0.34f, 0.2f);
            glVertex2f(0.2f, 0.6f);
    glEnd();


     glBegin(GL_POLYGON);            // These vertices form a closed polygon
            glColor3ub(35, 103, 31  );
            glVertex2f(0.25f, 0.2f);
            glVertex2f(0.15f, 0.2f);
            glVertex2f(0.0f, -0.0f);
    glVertex2f(0.4f, -0.0f);
    glEnd();

    glBegin(GL_POLYGON);            // These vertices form a closed polygon
            glColor3ub(85, 62, 43   );
            glVertex2f(0.22f, -0.0f);
            glVertex2f(0.18f, -0.0f);
            glVertex2f(0.18f, -0.4f);
    glVertex2f(0.22f, -0.4f);
    glEnd();

   glPopMatrix();
   glutTimerFunc(1500,disback,0);
   glFlush();
}

void display2(int val) {

 glutDisplayFunc(display3);



}
void display() {
   glClear(GL_COLOR_BUFFER_BIT);
   glLoadIdentity();
glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
glPushMatrix();

glBegin(GL_POLYGON);            // These vertices form a closed polygon
            glColor3ub(18, 62, 19 );
            glVertex2f(-0.99f, -0.2f);
            glVertex2f(0.99f, 0.0f);
```

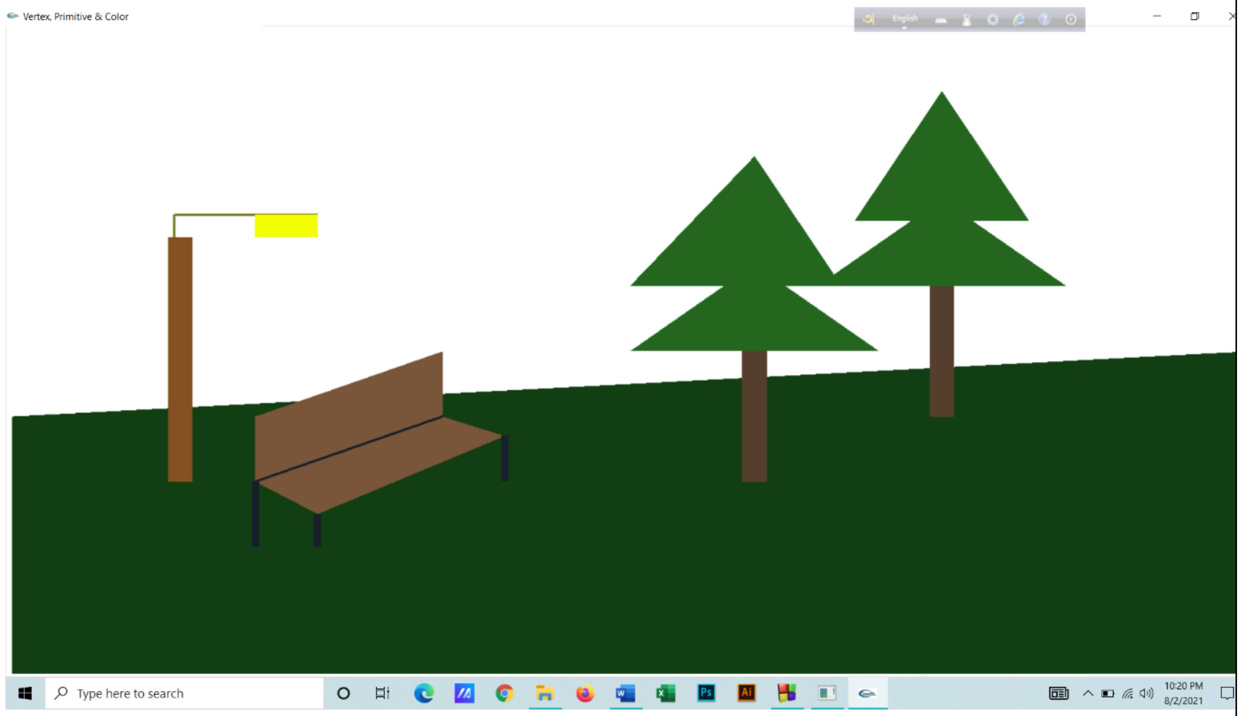
```
        glVertex2f(0.99f, -0.99f);
   glVertex2f(-0.99f, -0.99f);
   glEnd();
    glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(130, 80, 33  );
        glVertex2f(-0.74f, 0.35f);
        glVertex2f(-0.74f, -0.4f);
        glVertex2f(-0.7f, -0.4f);
   glVertex2f(-0.7f, 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();
```

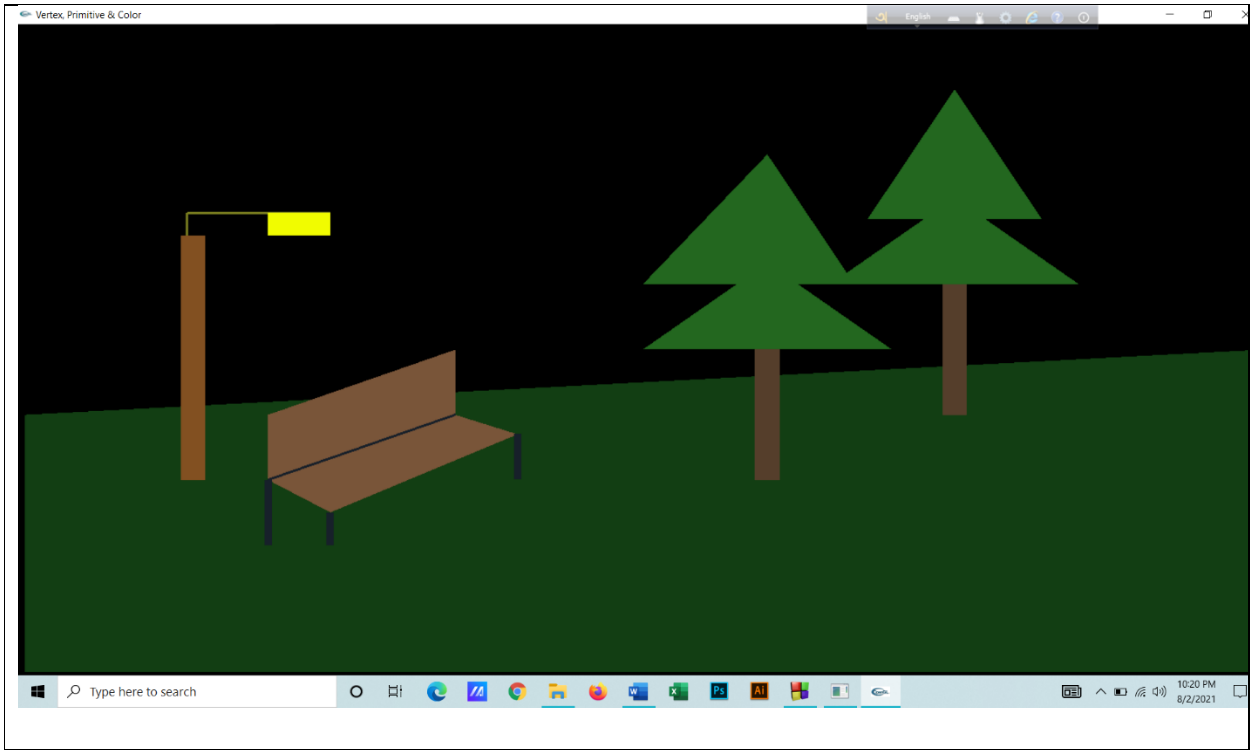
```
glLineWidth(3);
        // Draw a Red 1x1 Square centered at origin
        glBegin(GL_LINES); // Each set of 4 vertices form a quad
        glColor3ub(23, 32, 42 );
        glVertex2f(-0.3f, -0.2f);
        glVertex2f(-0.6f, -0.4f);
glEnd();


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(0.36f, 0.4f);
        glVertex2f(0.64f, 0.4f);
        glVertex2f(0.5f, 0.8f);
glEnd();


 glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(35, 103, 31  );
        glVertex2f(0.55f, 0.4f);
        glVertex2f(0.45f, 0.4f);
        glVertex2f(0.3f, 0.2f);
glVertex2f(0.7f, 0.2f);
glEnd();

glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(85, 62, 43   );
        glVertex2f(0.52f, -0.2f);
        glVertex2f(0.48f, -0.2f);
```

```
         glVertex2f(0.48f, 0.2f);
   glVertex2f(0.52f, 0.2f);
   glEnd();


      glBegin(GL_POLYGON);          // These vertices form a closed polygon
         glColor3ub(35, 103, 31  );
         glVertex2f(0.0f, 0.2f);
         glVertex2f(0.34f, 0.2f);
         glVertex2f(0.2f, 0.6f);
   glEnd();


    glBegin(GL_POLYGON);          // These vertices form a closed polygon
         glColor3ub(35, 103, 31  );
         glVertex2f(0.25f, 0.2f);
         glVertex2f(0.15f, 0.2f);
         glVertex2f(0.0f, -0.0f);
   glVertex2f(0.4f, -0.0f);
   glEnd();

   glBegin(GL_POLYGON);          // These vertices form a closed polygon
         glColor3ub(85, 62, 43   );
         glVertex2f(0.22f, -0.0f);
         glVertex2f(0.18f, -0.0f);
         glVertex2f(0.18f, -0.4f);
   glVertex2f(0.22f, -0.4f);
   glEnd();

glPopMatrix();

glutTimerFunc(1500,display2,0);
glFlush();

}

void dis()
{
     glutDisplayFunc(display);
}
```

```
/* 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

        initGL();                    // Our own OpenGL initialization
          glutDisplayFunc(dis);

  glutTimerFunc(20, update, 0);
   glutTimerFunc(20, update1, 0);



        glutMainLoop();              // Enter the event-processing loop
        return 0;
}
```

Output Screenshot (Full Screen)-





Question-
Create a simple day and night scenario using keyboard interaction. The key ‘D’ or ‘d’ will initiate the day mode and the key ‘N’ or ‘n’ will initiate the night mode.

Code-

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

void night();
void day();
/* 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(18, 62, 19 );
        glVertex2f(-0.99f, -0.2f);
        glVertex2f(0.99f, 0.0f);
        glVertex2f(0.99f, -0.99f);
    glVertex2f(-0.99f, -0.99f);
    glEnd();
     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);
```

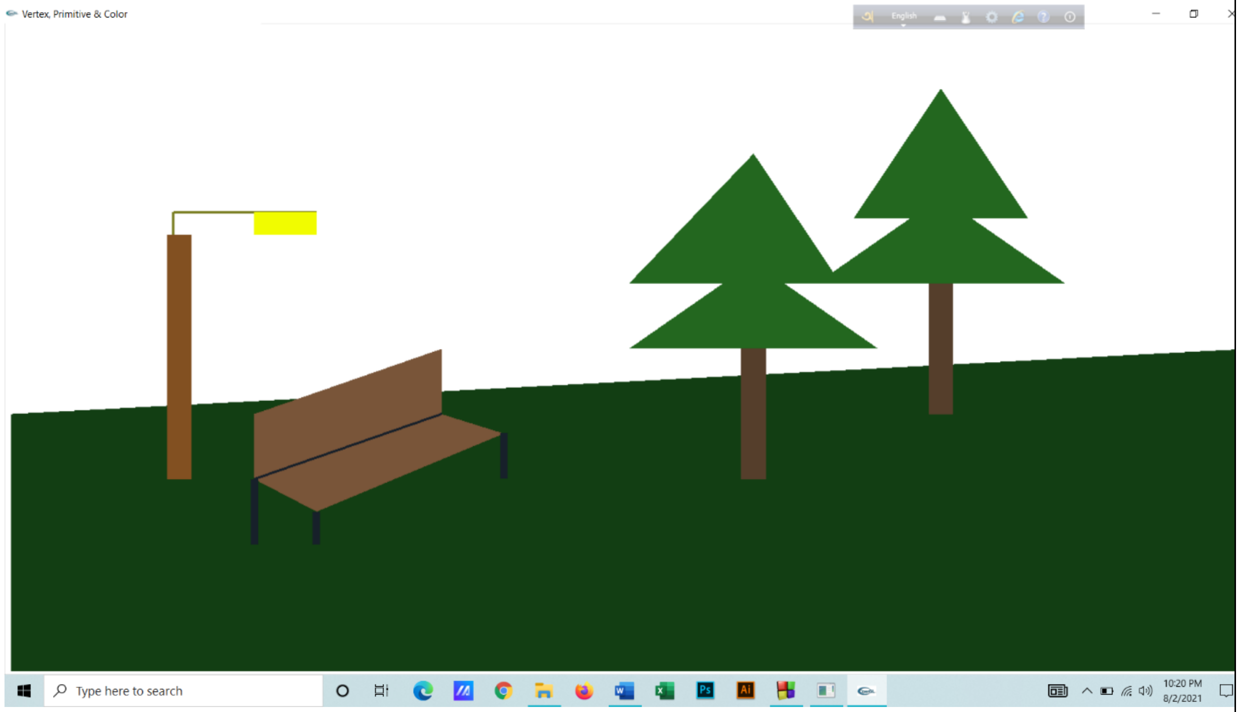
```
glVertex2f(-0.2f, -0.26f);
glEnd();

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(0.36f, 0.4f);
        glVertex2f(0.64f, 0.4f);
        glVertex2f(0.5f, 0.8f);
glEnd();


 glBegin(GL_POLYGON);           // These vertices form a closed polygon
        glColor3ub(35, 103, 31  );
        glVertex2f(0.55f, 0.4f);
        glVertex2f(0.45f, 0.4f);
        glVertex2f(0.3f, 0.2f);
glVertex2f(0.7f, 0.2f);
glEnd();

glBegin(GL_POLYGON);           // These vertices form a closed polygon
```

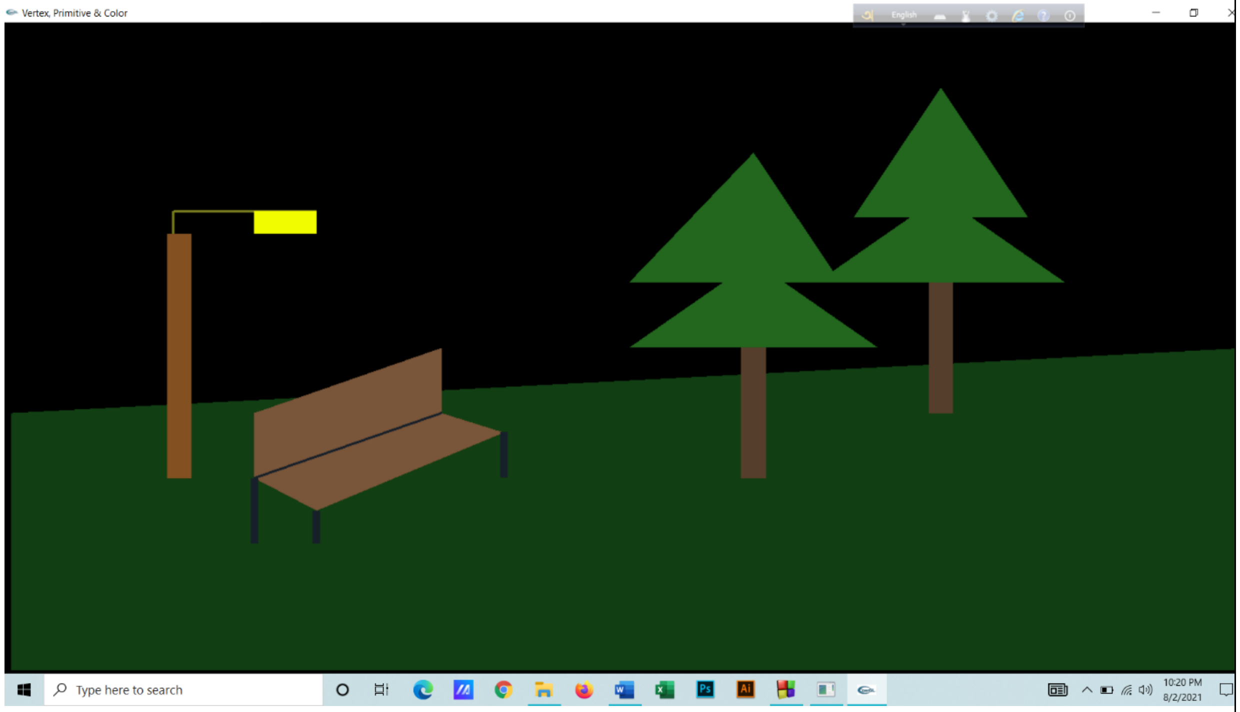
```
         glColor3ub(85, 62, 43   );
         glVertex2f(0.52f, -0.2f);
         glVertex2f(0.48f, -0.2f);
         glVertex2f(0.48f, 0.2f);
   glVertex2f(0.52f, 0.2f);
   glEnd();


      glBegin(GL_POLYGON);          // These vertices form a closed polygon
         glColor3ub(35, 103, 31  );
         glVertex2f(0.0f, 0.2f);
         glVertex2f(0.34f, 0.2f);
         glVertex2f(0.2f, 0.6f);
   glEnd();


    glBegin(GL_POLYGON);          // These vertices form a closed polygon
         glColor3ub(35, 103, 31  );
         glVertex2f(0.25f, 0.2f);
         glVertex2f(0.15f, 0.2f);
         glVertex2f(0.0f, -0.0f);
   glVertex2f(0.4f, -0.0f);
   glEnd();

   glBegin(GL_POLYGON);          // These vertices form a closed polygon
         glColor3ub(85, 62, 43   );
         glVertex2f(0.22f, -0.0f);
         glVertex2f(0.18f, -0.0f);
         glVertex2f(0.18f, -0.4f);
   glVertex2f(0.22f, -0.4f);
   glEnd();



         glFlush();  // Render now
}


void day()
{
   glClearColor(1.0,1.0,1.0,1.0);
glutPostRedisplay();

glFlush();
```

```
}
void night()
{
   glClearColor(0.0,0.0,0.0,1.0);
glutPostRedisplay();
}



void handleKeypress(unsigned char key, int x, int y) {

           switch (key) {

case 'd':

      glClearColor(1.0,1.0,1.0,1.0);
glutPostRedisplay();
  // glutDisplayFunc(day);
   //day();
   break;
case 'n':
 glClearColor(0.0,0.0,0.0,1.0);
glutPostRedisplay();
   break;





           }
}

/* 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
           glutKeyboardFunc(handleKeypress);

           glutMainLoop();               // Enter the event-processing loop
```

```
        return 0;
}
```

Output Screenshot (Full Screen)-


Vertex, Primitive & Color
Type here to search
10:20 PM
8/2/2021


Vertex, Primitive & Color
Type here to search
10:20 PM
8/2/2021