6. Program to draw a simple shaded scene consisting of a tea pot on a table. Define suitably the position and properties of the light source along with the properties of the properties of the surfaces of the solid object used in the scene.

```
#include<GL/glut.h>
void teapot(GLfloat x, GLfloat y, GLfloat z)
{
    glPushMatrix ();            //save the current state
    glTranslatef (x, y, z);     //move your item appropriately
    glutSolidTeapot (0.1);      //render your teapot
    glPopMatrix ();       //get back your state with the recent changes that you have done
}

void tableTop(GLfloat x, GLfloat y, GLfloat z)  // table top which is actually a CUBE
{
    glPushMatrix ();
    glTranslatef (x, y, z);
    glScalef (0.6, 0.02, 0.5);
    glutSolidCube (1);
    glPopMatrix ();
}
```

glPushMatrix — pushes the current matrix stack. There is a stack of matrices for each of the matrix modes. In GL_MODELVIEW mode, the stack depth is at least 32. In the other modes, GL_COLOR, GL_PROJECTION, and GL_TEXTURE, the depth is at least 2. The current matrix in any mode is the matrix on the top of the stack for that mode. glPushMatrix pushes the current matrix stack down by one, duplicating the current matrix. That is, after a glPushMatrix call, the matrix on top of the stack is identical to the one below it. glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack. Initially, each of the stacks contains one matrix, an identity matrix.

```
void tableLeg(GLfloat x, GLfloat y, GLfloat z)  // table leg which is actually a CUBE
{
    glPushMatrix ();
    glTranslatef (x, y, z);
    glScalef (0.02, 0.3, 0.02);
    glutSolidCube (1);
    glPopMatrix ();
}
```

glutSolidCube(size) and glutWireCube(size) render a solid or wireframe cube respectively. The cube is centered at the modeling coordinates' origin with sides of length size.

```
void wall(GLfloat x, GLfloat y, GLfloat z)   // wall which is actually a CUBE
{
    glPushMatrix ();
    glTranslatef (x, y, z);
    glScalef (1, 1, 0.02);
    glutSolidCube (1);
    glPopMatrix ();
}

void light()                  // set the lighting arrangements
{
    GLfloat mat_ambient[]  =  {1, 1, 1, 1};    // ambient colour
    GLfloat mat_diffuse[]  =  {0.5, 0.5, 0.5, 1};
    GLfloat mat_specular[]  =  {1, 1, 1, 1};
    GLfloat mat_shininess[]  =  {50.0f};        // shininess value
    glMaterialfv (GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv (GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv (GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv (GL_FRONT, GL_SHININESS, mat_shininess);

    GLfloat light_position[]  =  {2, 6, 3, 1};
    GLfloat light_intensity[]  =  {0.7, 0.7, 0.7, 1};

    glLightfv (GL_LIGHT0, GL_POSITION, light_position);
    glLightfv (GL_LIGHT0, GL_DIFFUSE, light_intensity);
}

void display()
{
    GLfloat teapotP = -0.07, tabletopP = -0.15, tablelegP = 0.2, wallP = 0.5;
    glClear (GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    gluLookAt (-2, 2, 5, 0, 0, 0, 0, 1, 0);  // camera position & viewing

    light ();                    //Adding light source to your project

    teapot (0, teapotP, 0); //Create teapot

    tableTop (0, tabletopP, 0); //Create table's top

    tableLeg (tablelegP, -0.3, tablelegP); //Create 1st leg
    tableLeg (-tablelegP, -0.3, tablelegP); //Create 2nd leg
    tableLeg (-tablelegP, -0.3, -tablelegP); //Create 3rd leg
    tableLeg (tablelegP, -0.3, -tablelegP); //Create 4th leg

    wall (0, 0, -wallP); //Create 1st wall
    glRotatef (90, 1, 0, 0);

    wall (0, 0, wallP); //Create 2nd wall
    glRotatef (90, 0, 1, 0);

    wall (0, 0, wallP); //Create 3rd wall

    glFlush ();     // show the output to the user
}
void init()
{
    glClearColor (0, 0, 0, 1);     // black colour background
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glOrtho (-1, 1, -1, 1, -1, 10);
    glMatrixMode (GL_MODELVIEW);
}
```

glMaterial — specify material parameters for the lighting model. fv means floating point vector

glMaterial takes three arguments. The first, face, specifies whether the GL_FRONT materials, the GL_BACK materials, or both GL_FRONT_AND_BACK materials will be modified. The second, pname, specifies which of several parameters in one or both sets will be modified. The third, params, specifies what value or values will be assigned to the specified parameter.

glLight sets the values of individual light source parameters. It takes 3 parameters – light, pname, params.

light names the light and is a symbolic name of the form GL_LIGHT i, where i ranges from 0 to the value of GL_MAX_LIGHTS -1.

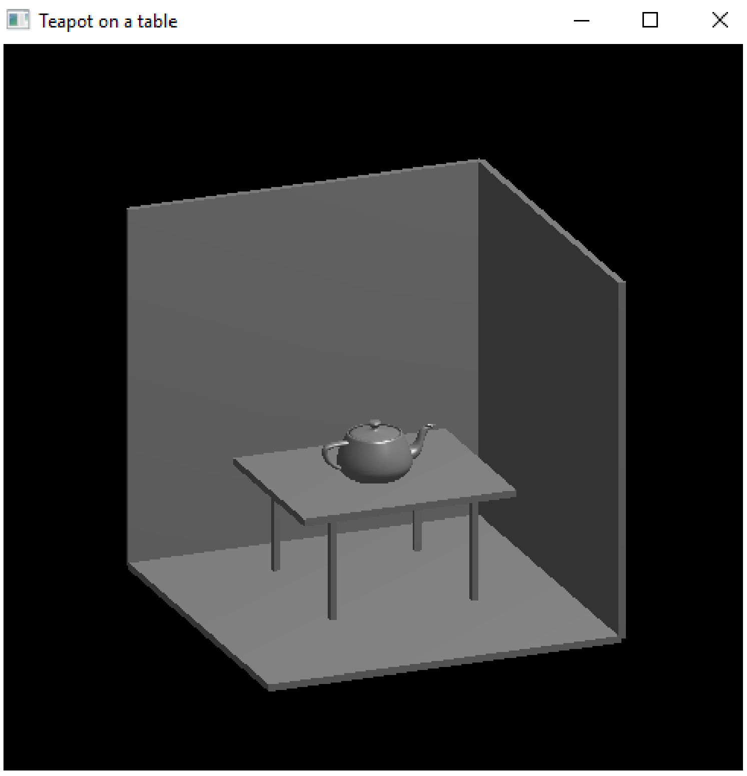
pname specifies one of ten light source parameters, again by symbolic name.

params is either a single value or a pointer to an array that contains the new values.

```
int main (int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Teapot on a table");

    init();

    glutDisplayFunc(display);

    glEnable(GL_LIGHTING);      // enable the lighting properties
    glEnable(GL_LIGHT0);        // enable the light source

    glShadeModel(GL_SMOOTH);  // for smooth shading (select flat or smooth shading)

    glEnable(GL_NORMALIZE);     // If enabled and no vertex shader is active, normal vectors
                                are normalized to unit length after transformation and before
                                lighting.
    glEnable(GL_DEPTH_TEST); // do depth comparisons and update the depth buffer.

    glutMainLoop();
}
```

****

<u>OUTPUT</u>


Teapot on a table