1. (10 pts) The surface of the equation \( x^2 + z^2 + y = 1 \) is called a *paraboloid*, shown in the picture below.

![Paraboloid Image]

It can be parameterized as

\[
\begin{align*}
    x &= t \sin s \\
    y &= 1 - t \\
    z &= t \cos s,
\end{align*}
\]

where \( 0 \leq s \leq 2\pi \) and \( 0 \leq t \leq 1 \). Letting \( P = (x, y, z) \), use the partial derivatives \( \frac{\partial P}{\partial s} \) and \( \frac{\partial P}{\partial t} \) to find a formula for the normal vector at \( P \).

2. (10 pts) In the previous problem, we could have chosen not to parameterize the equation. Instead, we could have written the equation as

\[
F(x, y, z) = x^2 + z^2 + y - 1 = 0
\]

and taken partial derivatives \( \frac{\partial F}{\partial x} \) and \( \frac{\partial F}{\partial z} \) to obtain the normal vectors. Using this approach to draw the paraboloid seen in the picture above, we would have to partition the intervals \([-1, 1]\) on the \( x \)- and \( z \)-axes to obtain the mesh. Discuss the disadvantages of using this approach for this particular surface.

3. (10 pts) In the *Mesh* class, for each vertex in the mesh, a normal vector is stored. The *Mesh* class could have been designed differently; it could have been designed so that a normal vector was stored for each facet, or quadrilateral, in the mesh. Upon rendering the mesh, what would be the visual effect of using one normal for each quadrilateral? Would this be desirable?
4. (10 pts) For each of the following parts, explain the essential difference between the two items, and then tell which is more efficient to compute and why.

(a) ambient reflection vs. diffuse reflection.
(b) positional light source vs. directional light source.
(c) local viewer vs. infinite viewer.

5. (10 pts) Assume the following:

- The light source is located at \((8.0, 0.0, 6.0)\).
- The diffuse light value is \((0.5, 0.5, 0.5)\).
- The diffuse material property of the surface at \((0.0, 0.0, 0.0)\) is \((1.0, 0.0, 0.0)\).
- The normal vector at \((0.0, 0.0, 0.0)\) is \((0.6, 0.8, 0.0)\).

Compute the diffuse reflection (RGB value) at the point \((0.0, 0.0, 0.0)\) on the surface and describe the color.

6. (10 pts) For each of the following transformations, draw a diagram that demonstrates what happens to the normal vectors if they undergo the same transformation that the points do. In each case, begin with the following drawing, which is in the \(xy\)-plane.

(a) The translation \(\text{glTranslatef}(2.0, 0.0, 0.0)\).
(b) The rotation \(\text{glRotatef}(90.0, 0.0, 0.0, 1.0)\).
(c) The scaling \(\text{glScalef}(3.0, 1.0, 1.0)\).

In which case(s) will the resulting normal vectors no longer be normal to the surface?

7. (10 pts) Explain the reason for each of the following design decisions in OpenGL.

(a) Performing lighting calculations in camera coordinates instead of clipping coordinates or normalized device coordinates.
(b) Performing clipping operations after projecting into clipping coordinates instead of in world or camera coordinates.

8. (10 pts) The projection matrix (for a perspective projection) is

\[
\begin{pmatrix}
\frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\
0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\
0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\
0 & 0 & -1 & 0
\end{pmatrix}
\]

The prototype of the function `glFrustum()` is

```c
void glFrustum(float l, float r, float b, float t, float n, float f);
```

For the function call `glFrustum(-4.0, 4.0, -3.0, 3.0, 1.0, 11.0)`, do the following.

(a) Write the projection matrix for this view frustum.
(b) Apply the projection matrix to the point (4.0, 3.0, -6.0, 1.0).
(c) Find the resulting point in normalized device coordinates after the perspective division.

9. (10 pts) When creating the view frustum, if we made `near` very small, say 0.00001, and made `far` very large, say 10000.0, what effect would this have on the objects being rendered once they were transformed into normalized device coordinates? Would this result in any undesirable visual effects? Explain? (Note: In the framebuffer, the depth coordinate (z) is stored with only 12-bit precision. Assume that this is the case in normalized device coordinates also.)