1. (a) Show that $a \times (b \times c) = b(a \cdot c) - c(a \cdot b)$ where $a, b, c \in \mathbb{R}^3$.
   (b) Can you give a geometric interpretation of this formula? (Hint: Consider the situation where a, b and c are co-planar and then generalize using linearity).
   (c) Suppose you have a known vector $a \in \mathbb{R}^3$ and an unknown vector v, but you know $a \times v$ and $a \cdot v$. Find v.
2. Let a, b be points in $\mathbb{R}^3$ and A, B be homogeneous encodings of these points (e.g. $A = (a, 1)$)
   (a) Show $A + B$ is the homogeneous coordinate of midpoint of line segment ab.
   (b) What is the $3D$ coordinate of $A + tB, t \geq 0$? Describe the shape of all such points.
3. Find the intersection of the ray $\mathbf{p} + t\mathbf{d}, t \geq 0$ with the ellipsoid $\mathbf{Au} + \mathbf{c}, \mathbf{u} \in \{(x, y, z) | x^2 + y^2 + z^2 \leq 1\}$. (You may use the function ϕ which maps a ray to its first intersection with the unit ball).
4. Suppose we want to render a scene with n triangles using a depth buffer (where we only compute the color of a pixel when the current depth is lower than the previous depth stored in the depth buffer).
   (a) What is the minimum number of color computations per pixel? (Give an example scene)
   (b) What is the maximum number of color computations per pixel? (Give an example scene)
   (c) Suppose we render the triangles in a random order. Prove that the expected number of color computations per pixel is $O(\log n)$.
5. What is the average degree of a vertex in a quad-mesh? (a mesh where every face is a quadrilateral) Hint: remember Euler's Formula:
   $$v - e + f = 2$$
6. Figure 2 shows GLSL fragment and vertex shaders for rendering an object that is lit by two directional light sources. There are at least 5 bugs in this code. Find as many bugs as you can and for each one, explain why it is a bug, what problem does it cause and what is the corrected version. (warning: incorrect bugs result in negative points!)
7. Remember in a Doubly Connected Edge List (DCEL), each half-edge stores a pointer to the next half edge of the same face in clockwise traversal order and a pointer to its twin half edge corresponding to a neighbouring face. Either prove that all $3D$ surfaces can be represented using DCELs or give an example surface that can not be represented by a DCEL.
8. Consider the following function:
   $$f(x) = (q(0, x)q^{-1}).v$$
   ($.v$ means the vector component of the quaternion) where $x \in \mathbb{R}^3$ and q is a unit quaternion.
   (a) Show that f is a linear function from $\mathbb{R}^3$ to $\mathbb{R}^3$.
   (b) Find the 3×3 matrix of f.
9. Find a formula for the control points of a Bézier curve with the following properties:
   - $f(0) = P_0$
   - $f(1) = P_1$
   - $f'(0) = D_0$
   - $f'(1) = D_1$
10. Design a data structure with $O(n \log^3 n)$ space comlpexity that supports $2D$ regular hexagonal range queries with base parallel to the x axis in $O(\log^3 n + k)$ time. (Figure 1) (Hint: Use a data structure similar to range trees)

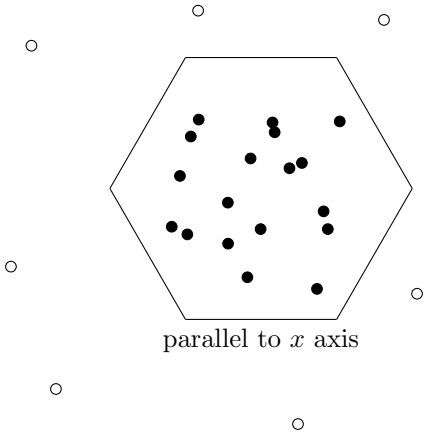

Figure 1: Hexagonal Range Query

```
1  // vertex shader
2  layout (location=0) in vec3 vertex_position;
3  layout (location=1) in vec3 vertex_normal;
4  layout (location=2) in vec2 vertex_uv;
5  out vec3 vs_normal_worldspace;
6  out vec2 vs_uv;
7  uniform mat4 model;
8  uniform mat4 view;
9  uniform mat4 projection;
10 void main(){
11     mat4 MVP = model * view * projection;
12     mat4 MV = model * view;
13     gl_Position = MVP * vec4(vertex_position, 1.0);
14     vs_normal_worldspace = (MV * vec4(vertex_normal, 1.0)).xyz;
15     vs_uv = vertex_uv;
16 }
17 // fragment shader
18 in vec3 vs_normal_worldspace;
19 in vec2 vs_uv;
20 uniform sampler2D diffuse_texture;
21 uniform vec3 light1_direction_worldspace;
22 uniform vec3 light1_color;
23 uniform vec3 light2_direction_worldspace;
24 uniform vec3 light2_color;
25 out vec3 color;
26 void main(){
27     vec3 p_texture = texture(diffuse_texture, vs_uv);
28    vec3 light1_contribution = dot(vs_normal_worldspace, light1_direction_worldspace) * p_texture *
          light1_color;
29    vec3 light2_contribution = dot(vs_normal_worldspace, light2_direction_worldspace) * p_texture *
          light2_color;
30    color = light1_contribution + light2_contribution;
31 }
```

Figure 2: GLSL code for Problem 6