Answer the questions in the spaces provided on the question sheets.

Name: ____________________________

Student Number: __________________

| Question 1 |  / 14 |
| Question 2 |  / 6  |
| Question 3 |  / 9  |
| Question 4 |  / 7  |
| Question 5 |  / 5  |
| TOTAL      |  / 41 |
This midterm has 5 questions, for a total of 41 points.

1. Culling and Clipping

The solid objects below are depicted in VCS, and the labels correspond to faces. Assume that during rendering the following tests are applied, in this specific order: view-frustum culling, clipping, back-face culling, z-buffer tests for fragments.

(a) (3 points) In alphabetical order, list each face as belonging to one of the culling categories below. If a face is partly visible, simply include it in the visible category.

- removed by view-frustum culling: \( r, s, u, v, w \)
- removed by clipping: \( f \)
- removed by back-face culling: \( c, d, f, h, j, n, g \)
- removed by z-buffer tests: \( e, g, i \)
- visible: \( a, b, j, m, o, p \)

(b) (1 point) True False If all the vertices are outside the view volume, then we can cull a triangle.

(c) (1 point) In NDCS, backfaces can be culled if this condition is true: \( N_z > 0 \).

Note: remember that NDCS has the z axis reversed, as compared to VCS, i.e., it is a left-handed coordinate system.

(d) (1 point) In graphics, we often compute the AABB for triangles or for entire objects. AABB is an acronym for: \( \text{axis-aligned bounding box} \)
(e) (3 points) Suppose we know three points that lie in a 3D plane: \( P_a, P_b, P_c \). Give the steps to compute an implicit plane equation. Is the plane equation unique?

\[
\vec{N} = (\vec{P}_b - \vec{P}_a) \times (\vec{P}_c - \vec{P}_a) \\
N = \langle A, B, C \rangle \\
F(P) = N \cdot P + D = 0 = Ax + By + Cz + D
\]

where \( D = -N \cdot P_a \)

\( = -N \cdot P_b = -N \cdot P_c \) Not unique; can scale by any constant.

(f) (1 point) A million-triangle object is outside of the view volume. Describe how we can efficiently avoid rendering it.

- build a bounding box (or bounding sphere)
- if all the vertices are outside with respect to a specific view volume plane, then we can cull

(g) (1 point) A million-triangle object is inside the view volume, but behind a solid wall. Describe how we can efficiently avoid rendering it.

- build a bounding box
- do a "virtual render" of this box
- if no pixels passed the z-buffer test then cull.

(h) (1 point) What is the advantage of using an *early z-buffer test*?

If the z-buffer test *fails*, then we can skip the calling the fragment shader \( \Rightarrow \) faster/more efficient

(i) (1 point) The `gl_Position` output of the vertex shader is in the \( \text{CCS} \) coordinate system (using an acronym is fine).

(j) (1 point) Clipping can be computed in the following coordinate frames (circle all that apply): \( \text{VCS}, \text{CCS}, \text{NDCS} \).
2. (6 points) Scan Conversion

Write pseudocode for scan converting the shape illustrated below, i.e., a square of dimensions $2r$, centered at $(x_c, y_c)$, with a circle removed from the inside. Use implicit equations to develop your solution. Assume that a pixel $(x, y)$ is set using \texttt{Set}$(x, y)$. If part of the object is located off-screen, the remainder should still be properly rendered. The result should be correct for any values of $x_c$, $y_c$, and $r$, and it should not try to scan convert pixels that are off-screen.

\[
\begin{align*}
Y_{\text{min}} &= \max(0, y_c - r) \\
X_{\text{max}} &= \min(h, x_c + r) \\
Y_{\text{min}} &= \max(0, y_c - r) \\
Y_{\text{max}} &= \min(h, y_c + r) \\
\text{for} \ (y = Y_{\text{min}}; \ y \leq Y_{\text{max}}; \ y++) \\
& \text{for} \ (x = X_{\text{min}}; \ x \leq X_{\text{max}}; \ x++) \\
F_L &= x - (x_c - r) \\
F_R &= (y_c + r) - x \\
F_T &= (y_c + r) - y \\
F_B &= y - (y_c - r) \\
F_C &= (x - x_c)^2 + (y - y_c)^2 - r^2 \quad \text{if} \ (\text{all } F \geq 0) \ \text{then set} \ (x, y)
\end{align*}
\]
3. Barycentric Coordinates

Assume that the barycentric coordinates are defined according to \( P = \alpha P_1 + \beta P_2 + \gamma P_3 \).

A given triangle is defined by \( P_1(0,70) \), \( P_2(30,10) \), \( P_3(70,50) \).

(a) (2 points) On the diagram above, sketch the triangle and label the vertices with their barycentric coordinates, e.g., \( P_1(\alpha, \beta, \gamma) \).

(b) (1 point) Sketch the three lines that correspond to \( \beta = 0 \), \( \beta = 0.5 \), \( \beta = 1 \).

(c) (2 points) Give an explicit equation for the line that corresponds to \( \beta = 0 \). Then give an implicit equation for the same line.

\[
y = 70 - \frac{2}{3}x \\
0 = -\frac{2}{3}x - y + 70 = \frac{F_{13}(x,y)}{F_{13}(x_1,y_1)} = \frac{1}{k}(-\frac{2}{3}x - y + 70)
\]

(d) (2 points) Develop an expression for \( \beta \), i.e., \( \beta = f(x,y) \). Do not bother with simplifying your expression.

\[
-\frac{2}{3}(30) - 10 + 70 = k \\
\beta = \frac{F_{13}(x,y)}{F_{13}(x_1,y_1)} = \frac{1}{k}(-\frac{2}{3}x - y + 70)
\]

(e) (2 points) Use barycentric interpolation to compute a value \( v \) at a point defined by \( \beta = 0.5, \gamma = 0.3 \), if the value of this quantity at the three vertices is given by: \( v_1 = 1, v_2 = 2, v_3 = 3 \).

\[
v = \alpha v_1 + \beta v_2 + \gamma v_3 \\
= 0.2(1) + 0.5(2) + 0.3(3) \\
= 0.2 + 1.0 + 0.9 \quad \Rightarrow \quad 2.1
\]
4. Texture Mapping

(a) (4 points) Consider the texture map below, which is to be mapped to Objects A and B. Assume that the RepeatWrapping texture mode is being used.

(i) In the Object A diagram above, sketch the image that would appear for Object A for the assigned texture coordinates.

(ii) In the Object B diagram above, assign texture coordinates to Object B so that it would yield the given image.

(b) (1 point) For the version of GLSL that we are using, shaders can have uniform, varying, attribute, and local variables. The (u,v) texture coordinates are passed into the vertex shader as ______ variables.

(c) (1 point) The (u,v) texture coordinates are passed into the fragment shaders as ______ variables.

(d) (1 point) T or F The computations needed for the perspective-correct barycentric interpolation of texture coordinates need to be coded as part of every fragment shader.
5. (5 points) Line-triangle intersection

Given a 3D line-segment, \( P_aP_b \), and a 3D triangle, \( P_1P_2P_3 \), determine how to compute the point at which the line intersects the plane of the triangle. Follow these steps: (a) write a parametric equation for a point on the 3D line segment; (b) write a parametric equation for a point in the plane of the triangle; (c) write your solution as a set of simultaneous linear equations that need to be solved.

(a) \( \mathbf{p}(t) = \mathbf{p}_a + t(\mathbf{p}_b - \mathbf{p}_a) \)

(b) \( \mathbf{p}(s,t) = \mathbf{p}_1 + s(\mathbf{p}_2 - \mathbf{p}_1) + t(\mathbf{p}_3 - \mathbf{p}_1) = \mathbf{p}_1 + s\mathbf{p}_2 + t\mathbf{p}_3 \)

(c) \( \mathbf{p}_a + t(\mathbf{p}_b - \mathbf{p}_a) = \mathbf{p}_1 + s(\mathbf{p}_2 - \mathbf{p}_1) + t(\mathbf{p}_3 - \mathbf{p}_1) \)

\( \mathbf{p}_a - \mathbf{p}_1 = -t(\mathbf{p}_b - \mathbf{p}_a) + s(\mathbf{p}_2 - \mathbf{p}_1) + t(\mathbf{p}_3 - \mathbf{p}_1) \)

\[ \begin{bmatrix} \mathbf{p}_a - \mathbf{p}_1 \\ \mathbf{p}_b - \mathbf{p}_a \\ \mathbf{p}_2 - \mathbf{p}_1 \\ \mathbf{p}_3 - \mathbf{p}_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix} \]

\( \Rightarrow \) solve for \( s, t, \gamma \)

\( \alpha = 1 - s - t \)

For a true intersection, we also require

\( s, t, \gamma \in [0, 1] \)

This is how fast ray-triangle intersections are computed for ray tracing.