1. (20 points) TRANSFORMS: Refer to the diagram below.

a. (10 pts) Write out a sequence of basic transformations (i.e. scale, rotate, translate) along with their parameters that accomplishes the transformation shown. Make sure that your ordering is clear. You will probably find it easier to use the 2D versions of the transformation matrices.

In multiplication order:

\[
T(-3, 3)\ R(90^\circ)\ S(-1, 1)\ T(-3, -3)
\]

\[
\begin{bmatrix}
\cos 90^\circ & -\sin 90^\circ & 0 \\
\sin 90^\circ & \cos 90^\circ & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
0 & -1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
0 & -1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
T = \text{translate} \\
R = \text{rotate} \\
S = \text{scale}
\]

b. (10 pts) Expand the basic transformations above into their matrix forms and multiply them together to get the final transformation matrix.

\[
\begin{bmatrix}
1 & 0 & -3 \\
0 & 1 & 3 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
0 & -1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
-1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]
2. (15 points) EULER ANGLES: Refer to the diagram below.

a. (10 pts) Recall that the transformation matrix for the common yaw-pitch-roll version of Euler angles is $R_{\text{yaw}}(\alpha)R_{\text{pitch}}(\beta)R_{\text{roll}}(\gamma)$, where the three axes of rotation are shown in the figure. Write out these rotation matrices in order, then multiply them together to get the general form.

Let $c\alpha = \cos(\alpha)$

$s\alpha = \sin(\alpha)$

etc...

$$
\begin{bmatrix}
  c\alpha & 0 & s\alpha \\
  0 & 1 & 0 \\
 -s\alpha & 0 & c\alpha 
\end{bmatrix}
\begin{bmatrix}
  1 & 0 & 0 \\
  0 & c\beta & s\beta \\
 -s\beta & c\beta & 1 
\end{bmatrix}
\begin{bmatrix}
  c\gamma & -s\gamma & 0 \\
  s\gamma & c\gamma & 0 \\
  0 & 0 & 1 
\end{bmatrix}
$$

$$
\begin{bmatrix}
  c\alpha c\beta & s\alpha c\beta & s\beta \\
  -s\alpha & c\alpha & 0 \\
 -c\alpha s\beta & -s\alpha s\beta & c\beta 
\end{bmatrix}
$$

b. (5 pts) What configurations will result in gimbal lock? Be specific, showing coordinate values for each relevant parameter. What rotational axis (in world space) is redundant in those configurations?

Gimbal lock occurs when the first and third rotation axes are aligned, which happens in this case when $\beta = \pm 90^\circ$.

The world $Y$ axis is made redundant.
3. (15 points) PARAMETRIC SURFACES: The standard surface parameterization for a bilinear patch is \( \mathbf{p}(u, v) = (1 - u)(1 - v)\mathbf{a} + (u)(1 - v)\mathbf{b} + (1 - u)(v)\mathbf{c} + (u)(v)\mathbf{d} \), with \((u, v) \in [0, 1]^2\). Plugging in 0 and 1 for \((u, v)\) quickly shows that \( \mathbf{p}(0, 0) = \mathbf{a}, \mathbf{p}(1, 0) = \mathbf{b}, \mathbf{p}(0, 1) = \mathbf{c}, \) and \( \mathbf{p}(1, 1) = \mathbf{d} \). Come up with another surface parameterization that rescales the domain from \([0, 1]^2\) to \([-1, 1]^2\). That is, write out an equation for \( \mathbf{p}'(u, v) \) such that \( \mathbf{p}'(-1, -1) = \mathbf{a}, \mathbf{p}'(1, -1) = \mathbf{b}, \mathbf{p}'(-1, 1) = \mathbf{c}, \) and \( \mathbf{p}'(1, -1) = \mathbf{d} \).

We want to remap \([-1, 1]\) to \([0, 1]\), then use those numbers as input to \(\mathbf{p}(u, v)\). That is:

\[
\mathbf{p}'(u, v) = \mathbf{p}\left(\frac{u+1}{2}, \frac{v+1}{2}\right)
\]

Expanding:

\[
\mathbf{p}'(u, v) = \left(1 - \frac{u+1}{2}\right)\left(1 - \frac{v+1}{2}\right)\mathbf{a} + \left(\frac{u+1}{2}\right)\left(1 - \frac{v+1}{2}\right)\mathbf{b} + \\
\left(1 - \frac{u+1}{2}\right)\left(\frac{v+1}{2}\right)\mathbf{c} + \left(\frac{u+1}{2}\right)\left(\frac{v+1}{2}\right)\mathbf{d}
\]
4. (15 points) BACKFACE CULLING: Given the eye point $E = (E_x, E_y, E_z)$, the lookat point $L = (L_x, L_y, L_z)$, and the three triangle vertices $A = (A_x, A_y, A_z)$, $B = (B_x, B_y, B_z)$, and $C = (C_x, C_y, C_z)$, write a procedure that will return true if a triangle is facing or parallel to the viewer, and false if it’s not. Assume that the triangle follows a counter-clockwise winding convention, such that if the points A, B, C appear to the viewer in CCW order, the viewer is looking at the front side of the triangle.

Use the dot product to see if $(E-L)$ and $N$ are pointed the same direction.

```cpp
bool front_facing (E, L, A, B, C) {
    N = (B-A) x (C-A)
    if (N \cdot (E-L) ≥ 0)
        return true;
    else
        return false;
}
```
5. **(15 points)** BÉZIER SPLINES: The spline below consists of two Bézier curves joined together at \( p_3 = q_0 \). The control points are arranged such that \( (p_3 - p_2) = (q_1 - q_0) \).

![Diagram of Bézier spline]

**a. (3 pts)** What level of continuity does the spline currently have? \( C^1 \)

**b. (6 pts)** Say that you move \( q_1 \) by +4 in the x coordinate, and +8 in the y coordinate. Assuming that \( q_1 \) and the endpoints \( p_0, p_3, q_2, \) and \( q_3 \) are held fixed, how would you move the remaining points to achieve \( C^1 \) continuity?

Move \( p_2 \) by \((-4, -8)\)

**c. (6 pts)** Starting with the same setup as part b, but holding the interior control points \( p_1, p_2, q_1, \) and \( q_2 \) fixed instead, how would you move the remaining points to achieve \( C^1 \) continuity?

Move \( q_0/p_3 \) by \((+2, +4)\)
6. (20 points) LINE CLIPPING: You are writing a function that will clip a line segment against a plane. Let the line segment be specified by the two endpoints $A = (A_x, A_y, A_z)$ and $B = (B_x, B_y, B_z)$, and let the plane be specified by the implicit equation $N \cdot x + d = 0$, where $N = (N_x, N_y, N_z)$ is the unit length plane normal. The picture below is just an example; $A$, $B$, $N$, and $d$ can be any arbitrary values.

![Line Clipping Diagram]

**a. (5 pts)** Write a function that will return true if a point is on the positive side of the plane ($N$ pointing towards it) or on the plane, and false if the point is on the negative side of the plane ($N$ pointing away). For example, it should return true for $A$ shown above, and false for $B$ shown above.

```python
bool side(N, d, A)
return (N \cdot A + d \geq 0)
```

**b. (10 pts)** Write a function that, assuming $A$ and $B$ are on opposite sides of the plane, will compute the point of intersection with the plane (p as shown in the figure). *(Hint: Create a parameterization of the line segment and plug that into the plane equation.)*

1. Start with:
   $p(t) = A + t(B - A)$
2. Solve for $t$:
   $N \cdot A + tN \cdot (B - A) + d = 0$
   $t = \frac{-d - N \cdot A}{N \cdot (B - A)}$
3. Then substitute into:
   $N \cdot p(t) + d = 0$
   Then plug $t$ into $p(t)$ to get the point of intersection.

**c. (5 pts)** Write a function that will take the two input points $A$ and $B$ and return the properly clipped line segment. You may call the functions in parts a and b as subroutines. Make sure to handle the cases when the entire segment stays, or is clipped.

```python
if (side(A) && side(B))
   return (A, B)
else if (!side(A) && side(B))
   return nothing
else if (side(A) && !side(B))
   else if (!side(A) && side(B))
      return (isect(A, B), B)
```