/* **1a**. Write down the GLM command that positions a camera at (0,0,10) looking straight down the z-axis. (5 pts.) */
glm::mat4 view = glm::translate(mat4(), vec3(0.0,0.0,10.0))

/* **1b**. Write down the GLM command to create a projection matrix that transforms a view frustum into clip space. Specifically, you want your projection to mimic naturalistic perspective, with a vertical field of view of 60 degrees and a depth of field that captures everything between 1 unit and 100 units away. Moreover you want your view to look good on a window that is 800x600 pixels. (5 pts.) */
glm::mat4 projection = glm::perspective(60.0f, 1.0f, 0.1f, 100.0f)

/* **1c**. Using that projection matrix and view matrix, describe each step of how a point would be placed at (2.5,2.5,-2) in 3D space and then transformed into a 2D pixel on your 800x600 window. That is, list the positions of the vertex at each stage of the rendering pipeline: In object coordinates, in model coordinates, in eye coordinates, in clip coordinates/normalized device coordinates, and finally in window coordinates. (30 pts.) */
Object Coordinates: fvec3(2.500000, 2.500000, -2.000000)
Model Coordinates: mat4x4((1.000000, 0.000000, 0.000000, 0.000000),
(0.000000, 1.000000, 0.000000, 0.000000),
(0.000000, 0.000000, 1.000000, 0.000000),
(2.500000, 2.500000, -2.000000, 1.000000))
Eye Coordinates: mat4x4((1.000000, 0.000000, 0.000000, 0.000000),
(0.000000, 1.000000, 0.000000, 0.000000),
(0.000000, 0.000000, 1.000000, 0.000000),
(2.500000, 2.500000, 8.000000, 1.000000))
Clip Coordinates: mat4x4((1.732051, 0.000000, 0.000000, 0.000000),
(0.000000, 1.732051, 0.000000, 0.000000),
(0.000000, 0.000000, -1.002002, -1.000000),
(4.330127, 4.330127, -8.216216, -8.000000))
Normalized Device Coordinates: fvec3(-0.541266, -0.541266, 1.027027)
Window Coordinates: fvec3(0.458734, 0.458734, 2.027027)

/* **2**. A cube is positioned at (2,2,-2) in 3D space. List the GLM commands to move the cube to (4,5,0), and that rotates the cube by 45 degrees around the x-axis and 45 degrees around the z-axis. Write down the 4x4 matrix for each of these operations individually, and then the 4x4 transformation matrix that concatenates all of these operations. (30 pts.) */
Cube model: mat4x4((1.000000, 0.000000, 0.000000, 0.000000),
(0.000000, 1.000000, 0.000000, 0.000000),
(0.000000, 0.000000, 1.000000, 0.000000),
(4.000000, 5.000000, 0.000000, 1.000000))
Rotate 45 degrees around the x-axis: mat4x4((1.000000, 0.000000, 0.000000, 0.000000),
(0.000000, 0.707107, 0.707107, 0.000000),
(0.000000, -0.707107, 0.707107, 0.000000),
(0.000000, 0.000000, 0.000000, 1.000000))
Rotate 45 degrees around the z-axis: mat4x4((0.707107, 0.707107, 0.000000, 0.000000),
(-0.707107, 0.707107, 0.000000, 0.000000),
(0.000000, 0.000000, 1.000000, 0.000000),
(0.000000, 0.000000, 0.000000, 1.000000))
Cube model concatenation after rotations and translation:

mat4x4((0.707107, 0.707107, 0.000000, 0.000000),
(-0.500000, 0.500000, 0.707107, 0.000000),
(0.500000, -0.500000, 0.707107, 0.000000),
(0.328427, 5.328427, 3.535534, 1.000000))