Student Number: **Name:**

Write down answers in-between questions. Please answer using short sentences.

The back of each page can be used for practice, but DO NOT write down the answer on the back.

Be sure to write your student number and name on each page.

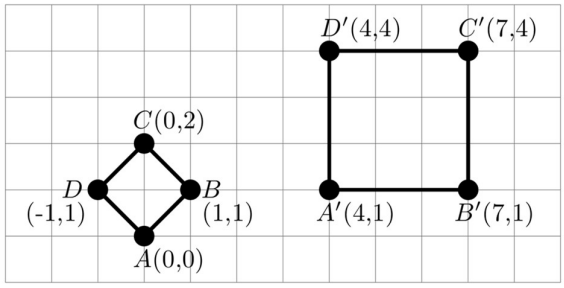
1. (5 pts) Match each output of rendering pipeline step of on the right with the corresponding rendering pipeline step on the left.

1) Vertex Processing	a) shaded fragments
2) Primitive Processing	b) primitives in screen space
3) Scan conversion	c) final output image
4) Fragment Processing	d) vertices in screen space
5) Per-sample operations	e) fragments

2. (4 pts) As a result of executing the following GLSL code, what is the value stored in the variable "result"?

```
mat3 m33 = mat3(1.0, 2.0, 3.0,
                4.0, 5.0, 6.0,
                7.0, 8.0, 9.0);
vec3 a = m33[1];
vec4 v4 = vec4(1.0, 2.0, 3.0, 4.0);
vec3 b = v4.xyy;
vec3 result = a * b;
```

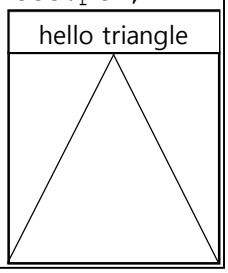
3. (7 pts) Give a matrix, or a product of matrices, which will transform the square ABCD to the square A'B'C'D'



Student Number: **Name:**

4. (7 pts) As in the code below, if you prepare a VAO and use it for rendering, a triangle is drawn as shown in the lower right corner (the square means the program window, and "hello world" is the window title).

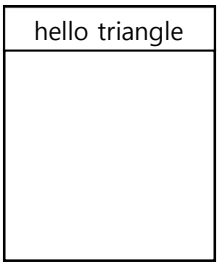
```
vertices = glm.array(glm.float32,       -1.0, -1.0, 0.0,
        1.0, -1.0, 0.0,         0.0,  1.0, 0.0    )
VAO = glGenVertexArrays(1); glBindVertexArray(VAO)
VBO = glGenBuffers(1);      glBindBuffer(GL_ARRAY_BUFFER, VBO)
glBufferData(GL_ARRAY_BUFFER, vertices.nbytes, vertices.ptr,
GL_STATIC_DRAW)
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE,
3*glm.sizeof(glm.float32), None)
glEnableVertexAttribArray(0)
...
glBindVertexArray(VAO)
glDrawArrays(GL_TRIANGLES, 0, 3)
...
```



Draw the resulting shape inside the "window" rectangle below if the third argument of glVertexAttribPointer() call in this code is changed as shown below.

```
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE,
2*glm.sizeof(glm.float32), None)
```

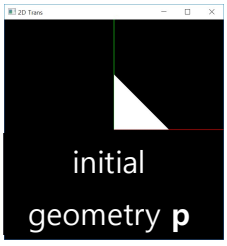
Answer:

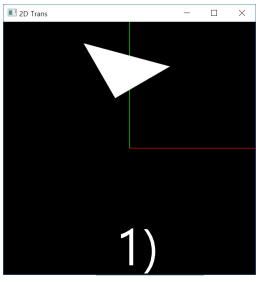


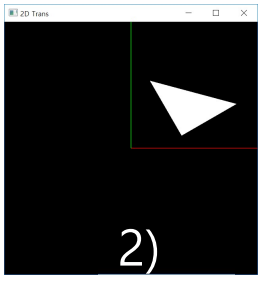
5. (5 pts) What is the result of the composite transformation **TFR** (**p' = TFRp**)? Note that the 2D spaces in the given figures range from -1 to 1 both for x and y directions.

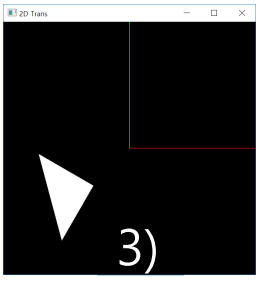
$\theta = 60$ degrees

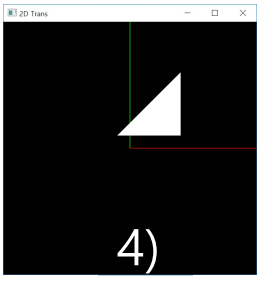
$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad F = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} 1 & 0 & 0.4 \\ 0 & 1 & 0.1 \\ 0 & 0 & 1.0 \end{bmatrix}$$











Student Number: **Name:**

6. (4 pts) In terms of coordinate-free geometric programming, choose **ALL undefined** operations for 3D points and vectors expressed in homogeneous coordinates.

   1) (2.1, 3.5, -10, 0) + (10, 5.3, 0, 0)

   2) (1, 0, 0, 1) + (0, 1, 0, 1)

   3) (0, -1, 2, 1) - (0, 0, 0, 0)

   4) 2.5 * (1, 0, 1, 1)

   5) (-5, -3, -1, 0) - (0, 0, 1, 0)

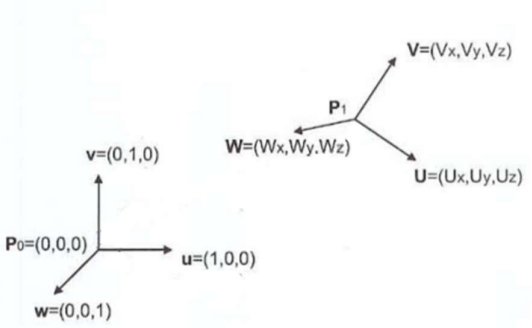
   6) (1, 1, 1, 1) - (1, 1, 1, 1)

7. (5 pts) Let's say the modeling, viewing, projection, viewport transformation matrices are $\mathbf{M_m}$, $\mathbf{M_v}$, $\mathbf{M_{pj}}$, $\mathbf{M_{vp}}$, respectively. Write down an equation describing the relationship between the position of a vertex in the object space, $\mathbf{p_o}$, and the position of the corresponding vertex in the screen space, $\mathbf{p_s}$ using given transformation matrices.

8. (6 pts) There are two affine transformations **T** and **S** on n-dimensional spaces. Their linear parts are represented by n x n matrix $\mathbf{M_t}$ and $\mathbf{M_s}$, and their translational parts are represented by n x 1 column vectors $\mathbf{t_t}$ and $\mathbf{t_s}$, respectively. Let's say you have a n-dimensional point **p**. What is the new point **p'** where the point **p** is transformed by the transformation **T compose S?** Fill in the blanks (a), (b), and (c).

   p' = __(a)__ p + __(b)__ + __(c)__

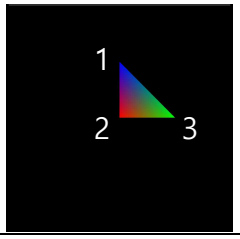
9. (6 pts) Suppose that we have two frames: the frame {0} has its origin at $\mathbf{P_0}$ and three standard bases **u**, **v**, **w**, and the frame {1} has its origin at $\mathbf{P_1}$=(P_x, P_y, P_z) and three orthonormal basis vectors **U**=(U_x, U_y, U_z), **V**=(V_x, V_y, V_z), **W**=(W_x, W_y, W_z). Write a 4x4 homogeneous matrix that transforms the position of a point with respect to the frame {1} into the position with respect to the frame{0}.



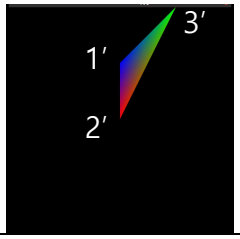
Student Number: **Name:**

10. (7 pts) In this OpenGL program, each vertex position specified as a vertex attribute is multiplied by the mat4 uniform variable M to calculate gl_Position. Setting M as shown on the left renders the figure as shown on the left. Fill in the blanks (a) and (b) in the code on the right so that the figure is drawn like the one on the right.

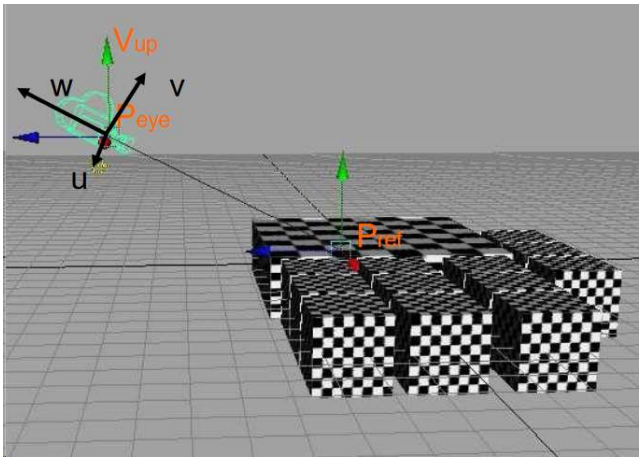
```
M = np.identity(4)
glUniformMatrix3fv(M_loc,    1,
GL_TRUE, M)
```




```
M = glm.mat4(__(a)__)
glUniformMatrix3fv(M_loc,    1,
__(b)__, M)
```



11. (7 pts) Express the three basis vectors **u**, **v**, **w** and the origin point **p** of the camera frame using the eye point $\mathbf{P}_{eye}$, look-at point $\mathbf{P}_{ref}$, and up vector $\mathbf{V}_{up}$.



12. (4 pts) Choose **ALL** the correct descriptions for orthographic projection.

    1) Straight lines are mapped to straight lines.

    2) Distances between all points are preserved.

    3) Far away objects appear the same size as closer ones.

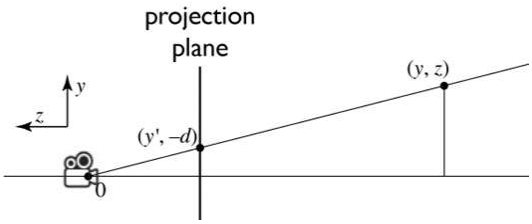
    4) Parallel lines are mapped to parallel lines.

    5) Angles are preserved.

Student Number: **Name:**

13. (6 pts) The transformation from the viewing frustum to a cuboid with near and far planes equal to those of the frustum is $\mathbf{P}_{f2c} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & f+n & fn \\ 0 & 0 & -1 & 0 \end{bmatrix}$, where n and f is the distances to near and far planes from the camera, respectively (that is, n>0 and f>0). If a point **p** is on the far plane of the viewing frustum and its x, y coordinates are (a, b) w.r.t. camera frame, what is the 3D position of **p** transformed by $\mathbf{P}_{f2c}$? Write down the position **after applying perspective division**. Assume that the camera is always looking in the -z direction in the camera space.

14. (7 pts) The figure below shows the perspective projection process that converts a 3D point (x, y, z) into a 2D point (x', y') on the projection plane. Fill in the blanks (a) to (g) in the formula below.



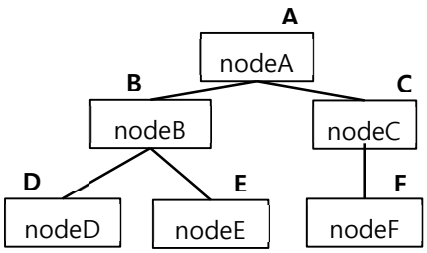
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \text{\_(a)\_} \\ \text{\_(b)\_} \\ 1 \end{bmatrix} \sim \begin{bmatrix} \text{\_(c)\_} \\ \text{\_(d)\_} \\ -z \end{bmatrix} = \begin{bmatrix} \text{\_(e)\_} & 0 & 0 & 0 \\ 0 & \text{\_(f)\_} & 0 & 0 \\ 0 & 0 & \text{\_(g)\_} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

15. (6 pts) In the following hierarchical model, A, B, …, F represents the local transformations (4x4 matrix) of nodeA, nodeB, …, nodeF, respectively.

    1) What is the nodeE's global transformation (E_g)?

    2) What the is nodeF's global transformation (F_g)?

    3) Fill in the blank: $E_g = F_g$ _____

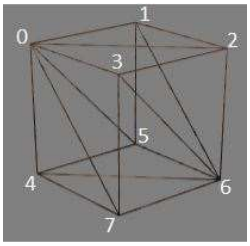


Student Number: **Name:**

16. (6 pts) Write the appropriate number in the blank (a), (b), and (c):

    When drawing a 3D cube (using 12 triangles) with the **separate triangles scheme**, **__(a)__** scalar numbers are included in the **vertex array**.

    When drawing a 3D cube (using 12 triangles) with the **indexed triangles scheme**, **__(b)__** scalar numbers are included in the **vertex array** and **__(c)__** scalar numbers are included in the **index array**.



17. (8 pts) The following Node class represents a node in a hierarchical model and builds a tree structure. Fill in the blank (a) and (b) to complete `update_tree_global_transform()` that calculates and updates the global transformation of a node and its all descendants.

```
class Node:
    def __init__(self, parent, shape_transform, color):
        self.parent = parent
        self.children = []
        if parent is not None:
            parent.children.append(self)

        self.transform = glm.mat4()  # local transform
        self.global_transform = glm.mat4()

        self.shape_transform = shape_transform
        self.color = color

    def set_transform(self, transform):
        self.transform = transform

    def update_tree_global_transform(self):
        if self.parent is not None:
            self.global_transform = __(a)__
        else:
            self.global_transform = self.transform

        for child in self.children:
            __(b)__
```