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) Choose **ALL** correct PyOpenGL function calls.

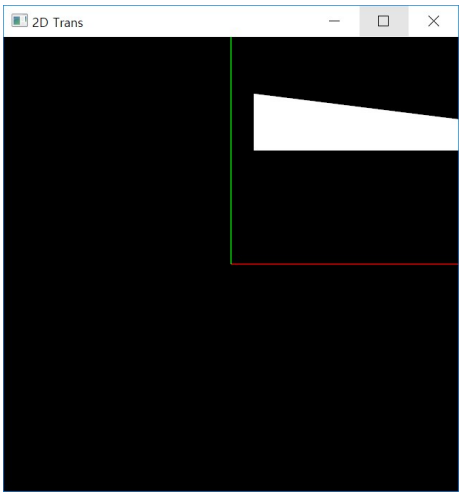
   1) glColor3ub(255, 0, 0, 0)

   2) glVertex2fv([-1.0,-1.0])

   3) glVertex2f((0.0, 1.0))

   4) glVertex2fv(np.array([1.0,-1.0])) # 'np' indicates the imported numpy module.

2. (3 pts) Which pair of the blank (a) and (b) render this image?




```
def render(M, u):
    glClear(GL_COLOR_BUFFER_BIT)
    glLoadIdentity()
    glBegin(GL_TRIANGLES)
    glColor3ub(255, 255, 255)
    glVertex2fv(M @ np.array([0.0,0.5]) + u)
    glVertex2fv(M @ np.array([0.0,0.0]) + u)
    glVertex2fv(M @ np.array([0.5,0.0]) + u)
    glEnd()
def main():
    # ...
    while not glfw.window_should_close(window):
        # ...
        M = np.array(__(a)__)
        u = np.array(__(b)__)
        render(M, u)
```

Student Number: **Name:**

1)
```
(a)
[[.5,0]
 [0,4]]
(b)
[.5,.1]
```

2)
```
(a)
[[2,0]
 [0,2]]
(b)
[.5,.1]
```

3)
```
(a)
[[4,0]
 [0,.5]]
(b)
[.1,.5]
```

4)
```
(a)
[[4,0]
 [0,.5]]
(b)
[.5,.5]
```

5)
```
(a)
[[4,1]
 [0,.5]]
(b)
[.1,.5]
```

3. (8 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. The question is:

   1) Let's say you have a n-dimensional point **p**. What is the new point **p'** from which the point **p** is transformed by the transformation **T compose S?** Fill in the blanks (a), (b), and (c).

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

4. (5 pts) Write down the 4 x 4 affine transformation matrix to rotate by an angle θ about the y-axis.

5. (9 pts) Let's say you want a triangle rendered by drawTriangle() to be transformed in the following order **w.r.t. global frame.** Fill in the blank (a), (b), and (c) to do this (You have to use glTranslatef(), glScalef(), and glRotatef() functions)

   1) First, translated by (1,0,2)
   2) Then, rotated 90 degrees about y axis
   3) Lastly, scaled by (-2, 2, 4)

```
def render():
    glClear(GL_COLOR_BUFFER_BIT)
    glLoadIdentity()

    ____(a)____
    ____(b)____
    ____(c)____

    drawTriangle()
```

Student Number: **Name:**

6. (5 pts) Choose **ALL** undefined operations in terms of coordinate-free geometric programming.

   1) (x1, y1, z1, 1) + (x2, y2, z2, 1)
   2) (x1, y1, z1, 0) + (x2, y2, z2, 0)
   3) (x1, y1, z1, 1) - (x2, y2, z2, 0)
   4) (x1, y1, z1, 0) - (x2, y2, z2, 0)
   5) (x1, y1, z1, 1) - (x2, y2, z2, 1)
   6) 2.5 * (x1, y1, z1, 1)

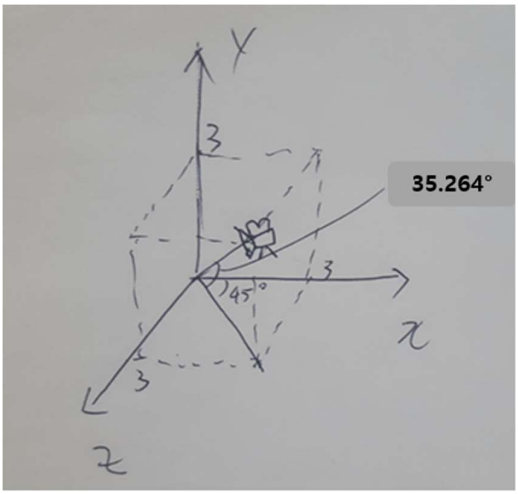
7. (5 pts) The position of point **p** is (5,-10,21) w.r.t. an affine frame $\mathbf{T}_1$, whose x-axis=(-1,0,0), y-axis=(0,1,0), z-axis=(0,0,1), and origin=(1,20,-5) w.r.t. global frame. Write down the position of **p** w.r.t. global frame.

8. (6 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.

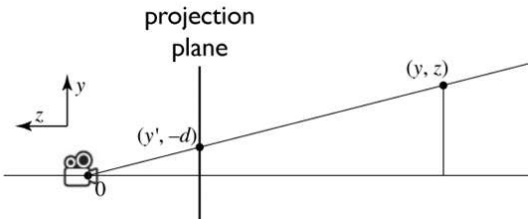
9. (9 pts) "moving camera" and "moving world" are essentially the same operation. Referring to the figure below, write a code that replaces the gluLookAt() function call in the render function below with two glRotatef() and one glTranslatef() function calls to show the same result as the render function below (If you need to calculate square root, you can use the np.sqrt() function).

```
def render():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glPolygonMode( GL_FRONT_AND_BACK, GL_LINE )
    glLoadIdentity()

    gluPerspective(45, 1, 1,10)

    gluLookAt(3,3,3, 0,0,0, 0,1,0)

    drawFrame()

    glColor3ub(255, 255, 255)
    drawCubeArray()
```

Student Number: **Name:**



10. (6 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.



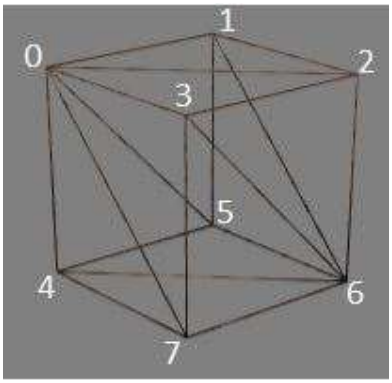
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \text{\_(a)\_} \\ \text{\_(b)\_} \\ 1 \end{bmatrix} \sim \begin{bmatrix} \text{\_(c)\_} \\ \text{\_(d)\_} \\ -z \end{bmatrix} = \begin{bmatrix} \text{\_(e)\_} & 0 & 0 & 0 \\ 0 & \text{\_(f)\_} & 0 & 0 \\ 0 & 0 & \text{\_(g)\_} & 0 \end{bmatrix}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

11. (3 pts) Complete the code below to render a single quad polygon by filling in the blanks (a), (b), and (c). Answers must be written in numbers (scalar), and answers in python expression format such as python variable names will be treated as incorrect answers.

```
varr = np.array([
            (  0 ,  0 ), # v0
            ( .5 ,  0 ), # v1
            ( .5 , .5 ), # v2
            (  0 , .5 ), # v3
            ], 'float32')
iarr = np.array([
          (0,1,2,3),
          ])
glVertexPointer((a), GL_FLOAT, (b), varr)
glDrawElements(GL_QUADS, (c), GL_UNSIGNED_INT, iarr)
```

Student Number: **Name:**

12. (6 pts) The following figure shows vertex indices of a cube object which is a triangle mesh. The following code creates an index array to draw this cube object with the glDrawElements () function. Fill in the blanks (a) and (b) below to complete the index array (use "counterclockwise" for the vertex winding order to indicate "front" face).




```
iarr = np.array([
            (0,2,1),
            (0,3,2),
            (4,5,6),
            (4,6,7),
            (0,1,5),
             ___(a)___,
            (3,6,2),
            (3,7,6),
            (1,2,6),
            (1,6,5),
            (0,7,3),
            ___(b)___,
            ])
```

13. (6 pts) Choose **ALL false(incorrect)** statement about the Phong illumination model.

    1) Phong illumination model is composed of three components; ambient, diffuse, specular component.

    2) The ambient component provides constant global lighting, which is a severe approximation of indirect lighting.

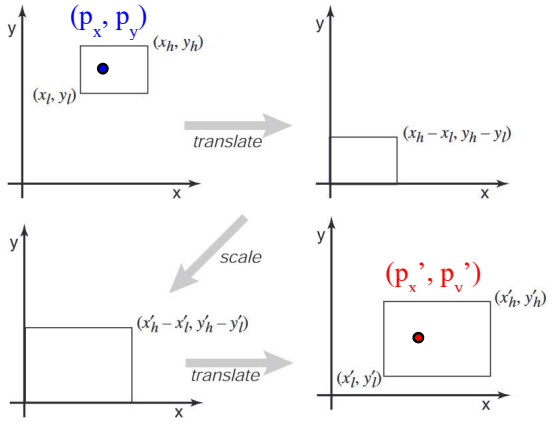
    3) The diffuse component assumes that light is reflected equally in all directions.

    4) The intensity of the diffuse component increases as the angle between the surface normal vector and the eye direction vector decreases.

    5) The specular component simulates highlights on shiny objects in a physically correct way.

    6) The intensity of the specular component increases as the angle between the surface normal vector and the light direction vector decreases.

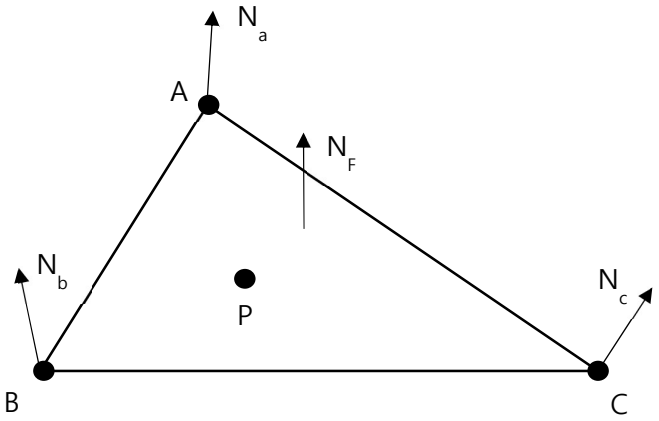
14. (12 pts) The following figure and equation show the process of getting a "windowing" transformation matrix that maps a point (p_x, p_y) in a rectangular space from (x_l, y_l) to (x_h, y_h) to a point (p_x', p_y') in a rectangular space from (x_l', y_l') to (x_h', y_h'). Fill in the blank (a) – (f) to complete the equation.



Student Number: **Name:**

$$\begin{pmatrix} p_x' \\ p_y' \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & (a) \\ 0 & 1 & (b) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} (c) & 0 & 0 \\ 0 & (d) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & (e) \\ 0 & 1 & (f) \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix}$$

15. (12 pts) You're about to render a triangle ABC which is one of triangles in a triangle mesh model. N_a, N_b, N_c are the vertex normal of the vertex A, B, C and calculated as the average of the face normals of the neighboring triangles. N_F is the face normal of triangle ABC.



The illumination model used for this rendering is given as a function `illum(p, n)` where p is the position of the target point to be illuminated and n is the surface normal at p to be used for illumination. For example, the illuminated color at the vertex A with N_a as surface normal is `illum(A, Na)`.

Question: What is the illuminated color at the point P in the triangle ABC when using…

1) Flat shading
2) Gouraud shading
3) Phong shading

Note that the barycentric coordinate of the interior point P is given as (u, v, w), meaning that P = uA + vB + wC. You have to use u, v, w to compute interpolated vertex color or vertex normal at P.

Note that A, B, C, and P mean both the name of vertices (or points) and their positions.