As we have discussed in lectures, the scaling, moving, and rotation matrices are:

\[
S(s_x, s_y) = \begin{bmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & 1
\end{bmatrix} \quad M(m_x, m_y) = \begin{bmatrix}
  1 & 0 & m_x \\
  0 & 1 & m_y \\
  0 & 0 & 1
\end{bmatrix} \quad R(\theta) = \begin{bmatrix}
  \cos \theta & -\sin \theta & 0 \\
  \sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
\end{bmatrix}
\]

It is ok for your answers to contain \( S(x,y) \), or \( M(x, y) \), or \( R(\theta) \).

**Problem 1. (30 Points)**

**Event driven programming.** Your new manager has decided you should implement a 2D application interacting with users to collect vertices for a polygon. Furthermore, he has decided that you have to base your development on the *StoogesGUI (SGUI)* library. In SGUI library, there is one call back for servicing mouse events:

```cpp
MouseService(TbuttonEnum WhichButton, TactionEnum WhatAction, int x, int y);
// TbuttonEnum: can be LeftMouseBotton, RightMouseButton, or NoMouseButton
// TactionEnum: can be ButtonDown - mouse button went down
// ButtonUp - mouse button went up
// MotionWithButtonDown - mouse moving with button down
// or MotionWithButtonUp - mouse moving without any button depressed
// x and y are the current mouse position in device coordinate where lower-left corner is
// the origin.
```

With the above service function, you are to support polygon definition protocol:

1. Left mouse down begins the *vertex collection process* (VCP).
2. Thereafter, each left mouse down indicates another vertex to be collected.
3. During VCP, you should echo a line connecting the last defined vertex to the current mouse position *regardless of the left mouse state*. That is, during VCP, you should echo the line even when left mouse button is up.
4. Right mouse down indicates the end of VCP. At this point, all vertices are collected and you should complete the polygon definition by duplicating the 0-th vertex.

**Problem 1a: (15 Points)**. Design a state transition diagram supporting the above specified polygon definition protocol.

**Problem 1b: (15 Points)**. Design a C++ class that to implement your state transition diagram.
Problem 2. (70 Points)

2D Transformation and Hierarchical Modeling.
Refer to the following figures. The lower one shows the layout of the new CSSIE450 Lab, and the upper figures illustrate the dimensions of the furniture and the machines in the new lab.

If the CSSIE 450 Lab is modeled as follows:
Problem 2a: (16 Points). Calculate the details of each of the $T_1$, $T_2$, $T_3$, $T_4$ transformation matrices. Express your answer in terms of the given scaling ($S$), moving ($M$), and rotation ($R$) matrices. Make sure you define the parameters in each $S/M/R$ operator (define the $x$, $y$, and theta). You do not need to compute the concatenation of the matrices results.

Problem 2b: (10 Points). Here are the definitions of: DrawPC() and DrawDeskChair():

```cpp
DrawPC() {
    static Outer[4][2] = {{0.0, 0.0}, {3.0, 0.0}, {3.0, 2.0}, {0.0, 2.0}};
    static Inner[4][2] = {{0.2, 0.2}, {2.8, 0.2}, {2.8, 1.8}, {0.2, 1.8}};
    glBegin(GL_POLYGON);
        for (int I = 0; I < 4; I++)
            glVertex(Outer[I][0], Outer[I][1]);
    glEnd();
    glBegin(GL_POLYGON);
        for (I = 0; I < 4; I++)
            glVertex(Inner[I][0], Inner[I][1]);
    glEnd();
}

DrawDeskChair() {
    static Chair[4][2] = {{4.0, 1.0}, {6.0, 1.0}, {6.0, 4.0}, {4.0, 6.0}};
    static Desk[4][2] = {{0.0, 5.0}, {10.0, 5.0}, {10.0, 10.0}, {5.0, 10.0}};
    glBegin(GL_POLYGON);
        for (int I = 0; I < 4; I++)
            glVertex(Chair[I][0], Chair[I][1]);
    glEnd();
    glBegin(GL_POLYGON);
        for (I = 0; I < 4; I++)
            glVertex(Desk[I][0], Desk[I][1]);
    glEnd();
}
```

You are told to use your solutions to Problem 2a to implement Draw450Lab() by manipulating the GL matrix stack and calling the above two functions. Please show the C++ code for Draw450Lab(). Please clearly indicate which matrix stack you are using, and your assumptions (if any) on the content of the top of the matrix stack you are using.
**Problem 2c: (14 Points).** We know that the OpenGL `glOrtho2D()` and `glViewport()` functions define a matrix that transforms points in the defined World Coordinate (WC) Window to the Device Coordinate (DC) system. We also know that:

\[
\begin{align*}
glViewport(lowerLeftX, lowerLeftY, width, height) \\
gluOrtho2D(left, right, bottom, top)
\end{align*}
\]

Given that the drawable size is 600x600, and with the following OpenGL/function calls:

\[
\begin{align*}
glViewport(40, 60, 200, 300) \\
gluOrtho2D(26.0, 46.0, 1.0, 31.0) \\
Draw450Lab()
\end{align*}
\]

Please derive the World Coordinate to Device Coordinate mapping function.

**Problem 2d: (15 Points).** Please sketch the appearance of the drawable that would result from the code given in **Problem 2c**. Clearly indicate the coordinate/location of the viewport and clearly show the dimension/location of visible portion of **YourStation** (not **MyStation**), in Device Coordinate Units.

**Problem 2e: (15 Points).** Here is a really simple function, `DrawUnitSquare()`:

```java
DrawUnitSquare() {
    static UnitSquare[4][2] = {{0.0, 0.0}, {1.0, 0.0}, {1.0, 1.0}, {0.0, 1.0}};
    glBegin(GL_POLYGON);
        for (int I=0; I<4; I++)
            glVertex(UnitSquare[I][0], UnitSquare[I][1]);
    glEnd();
}
```

Show how to implement `DrawPC()` and `DrawDeskChair()` by using the GL matrix stack and calling the above function (multiple times). In this case, your `DrawPC()` and `DrawDeskChair()` functions cannot (and do not need to) call `glBegin()`.

This assignment contributes 5% towards your final grade for this course.