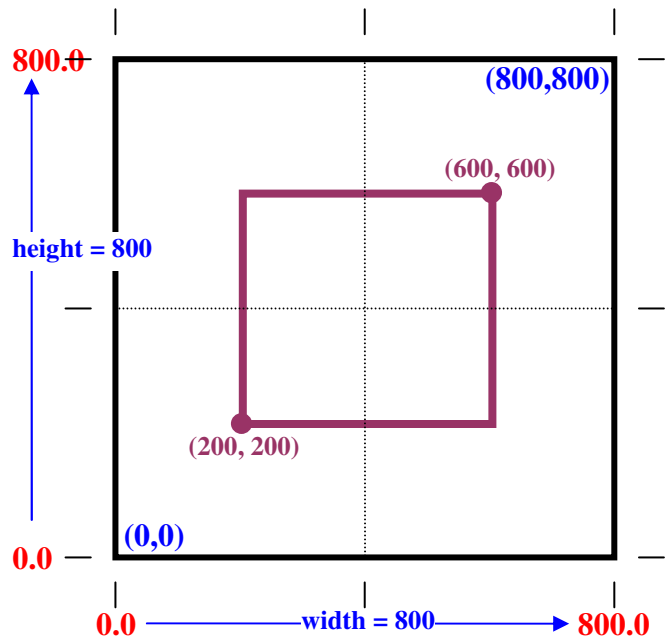
Illustrate what is drawn by the following display callbacks. Treat each case as appearing in a separate OpenGL/GLUT program. In all cases, assume an 800 x 800 window has been opened by GLUT.

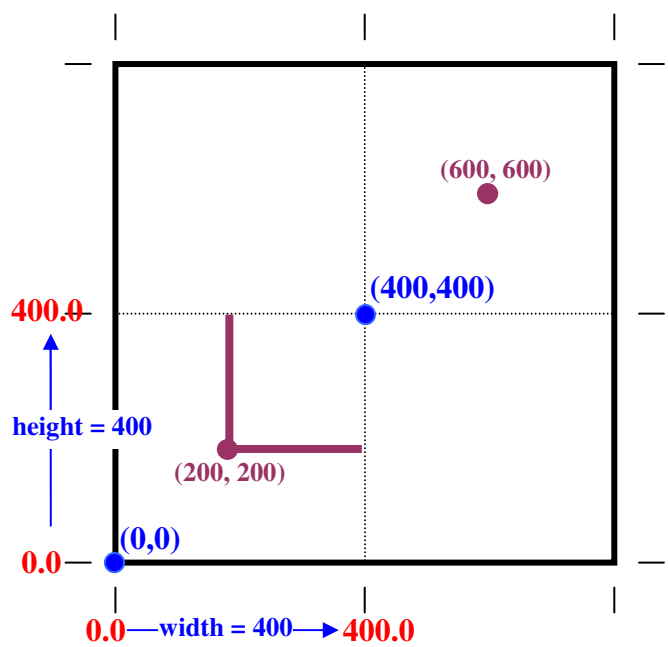
1.

```
void display( void )
{
  glClearColor(1.0, 1.0, 1.0, 1.0);
  glClear( GL_COLOR_BUFFER_BIT );
  glPolygonMode(GL_FRONT, GL_LINE);

  glViewport(0,0,800,800);
  glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluOrtho2D(0.0, 800.0, 0.0, 800.0);
  glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
  glColor3f(0.0, 0.0, 0.0);
  glRecti(200,200,600,600);

  glutSwapBuffers();
}
```



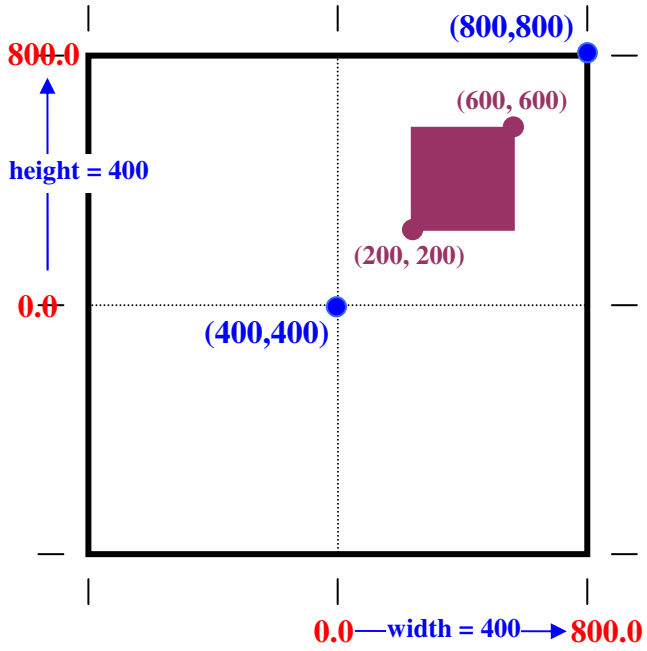
2.

```
void display( void )
{
  glClearColor(1.0, 1.0, 1.0, 1.0);
  glClear( GL_COLOR_BUFFER_BIT );
  glPolygonMode(GL_FRONT, GL_LINE);

  glViewport(0,0,400,400);
  glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluOrtho2D(0.0, 400.0, 0.0, 400.0);
  glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
  glColor3f(0.0, 0.0, 0.0);
  glRecti(200,200,600,600);

  glutSwapBuffers();
}
```



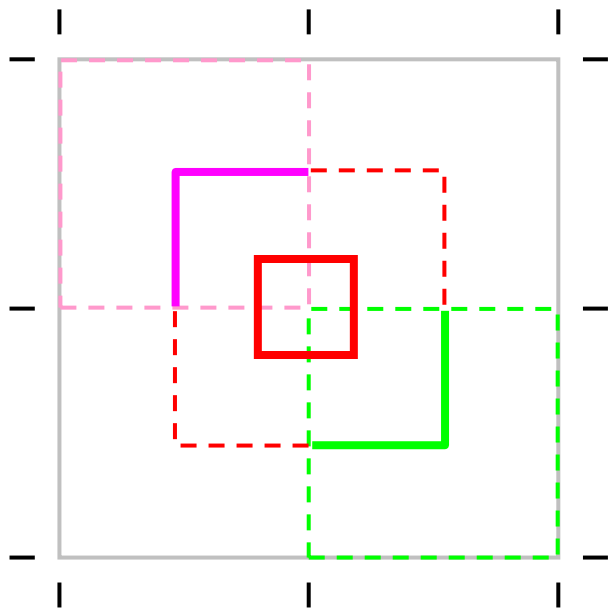
3.

```
void display( void )
{
  glClearColor(1.0, 1.0, 1.0, 1.0);
  glClear( GL_COLOR_BUFFER_BIT );
  glPolygonMode(GL_FRONT, GL_FILL);

  glViewport(400,400,400,400);
  glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluOrtho2D(0.0, 800.0, 0.0, 800.0);
  glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
  glColor3f(0.0, 0.0, 0.0);
  glRecti(200,200,600,600);

  glutSwapBuffers();
}
```

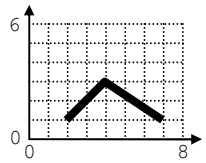


4.

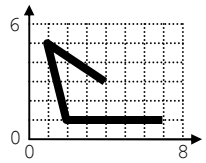
```
void display( void )
{
  glClearColor(1.0, 1.0, 1.0, 1.0);
  glClear( GL_COLOR_BUFFER_BIT );
  glPolygonMode(GL_FRONT, GL_LINE);

  glViewport(0,400,400,400);
  glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluOrtho2D(0.0, 400.0, 400.0, 800.0);
  glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
  glColor3f(0.0, 0.0, 0.0);
  glRecti(200,300,500,600);

  glViewport(400,0,400,400);
  glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluOrtho2D(400.0, 800.0, 0.0, 400.0);
  glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
  glColor3f(0.0, 0.0, 0.0);
  glRecti(300,200,600,500);

  glViewport(200,200,400,400);
  glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluOrtho2D(0.0, 1.0, 0.0, 1.0);
  glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    glTranslatef(0.5, 0.5, 0.0);
    glScalef(0.5, 0.5, 1.0);
  glColor3f(0.0, 0.0, 0.0);
  glRectf(-0.5, -0.5, 0.5, 0.5);

  glutSwapBuffers();
}
```



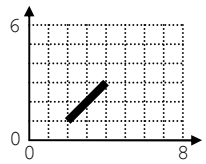
5. Match the drawing on the left with the code on the right that draws it (everything might not match!).



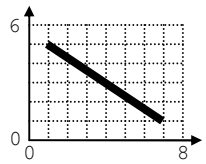
Answer: **C**



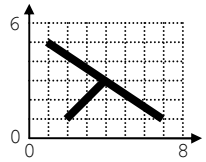
Answer: **H**



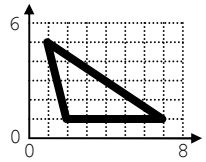
Answer: **A**



Answer: **H**



Answer: **D**



Answer: **G**

```
A. glBegin( GL_LINES );
     glVertex2i(2,1);
     glVertex2i(4,3);
     glVertex2i(7,1);
  glEnd();

B. glBegin( GL_LINES );
     glVertex2i(2,1);
     glVertex2i(7,1);
     glVertex2i(1,5);
   glEnd();

C. glBegin(GL_LINE_STRIP );
     glVertex2i(2,1);
     glVertex2i(4,3);
     glVeretx2i(7,1);
   glEnd();

D. glBegin( GL_LINE_STRIP );
     glVertex2i(2,1);
     glVertex2i(4,3);
     glVertex2i(7,1);
     glVertex2i(1,5);
   glEnd();

E. glBegin( GL_LINE_LOOP );
     glVertex2i(2,1);
     glVertex2i(4,3);
     glVertex2i(7,1);
     glVertex2i(1,5);
   glEnd();

F. glBegin( GL_LINE_LOOP );
     glVertex2i(4,3);
     glVertex2i(2,1);
     glVertex2i(7,1);
     glVertex2i(1,5);
   glEnd();

G. glBegin( GL_POLYGON );
     glVertex2i(4,3);
     glVertex2i(1,5);
     glVertex2i(2,1);
     glVertex2i(7,1);
   glEnd();

H. (no match)
```

6. Explain the concept of "double-buffering".

Changing the contents of the framebuffer during a refresh of the raster causes flickering. To avoid this, we logically divide the framebuffer bitplanes into two parts, treating half of the bitplanes as a "back" buffer, and the other half as a "front"buffer. The front buffer is always displayed but never drawn into; we always draw into the back buffer but never display it. After drawing into the back buffer is completed, we "swap" the two buffers: front becoming back, back becoming front. This is the typical way of doing animation.

7. Explain how stroke text is positioned in the viewport. **The same way other geometry is positioned: glVertex* transformed by using rotations, translations and scales.**

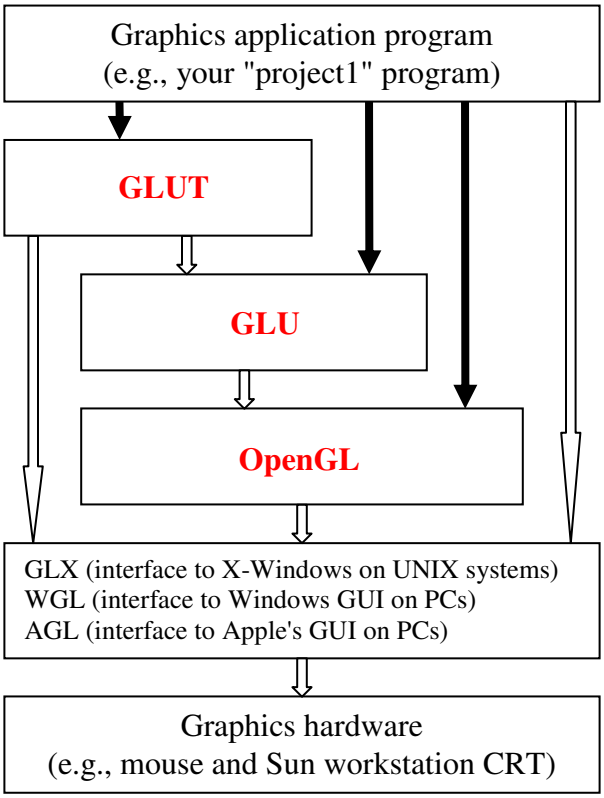
8. OpenGL can be viewed as a "state machine". List two things that are part of the OpenGL state, and give an example of changing the OpenGL state.

color, raster position, line width, line stipple pattern, viewport, …

glEnable, glDisable, or another function such as glColor*, glLineWidth, ...

9. We are using several application programming interfaces (API): **OpenGL, GLU, GLUT**

Place each of these API in its correct spot in the diagram below. The solid arrows correspond to API functions calls that you have made in your graphics programs in this class (e.g., `glBegin`, `gluOrtho2D`, `glutSwapbuffers`).



10. Explain how using GLUT is a good example of "event-driven programming"

An event-driven program essentially sits idle, in an <u>infinite loop</u>, waiting <u>events</u> to occur. Events can be such things as a <u>mouse movement, keypress, mouse button press</u>, display needing to be updated, menu selection, etc. When an event occurs its <u>event-handler</u> is called.

To use GLUT one has to register a <u>callback</u> as an event handler for each type of event one wishes to handle. A program that uses GLUT enters an infinite loop (glutMainLoop). During this loop, GLUT looks to see if an event has occured. When an event occurs, the corresponding callback function is executed.

11. Your OpenGL/GLUT program compiles, a window opens, but none of the things you're trying to render appear. It's a simple program: double-buffered, just drawing some geometry, and animating that geometry using transformations. You check and sure enough: `glutInitDisplayMode()` is correct, the display callback is registered, and you didn't forget to actually draw something in the display callback. Give five (3) explanations as to what might be wrong with your code. Be specific.

1. Drawing in the same color used to clear the framebuffer.
2. Clearing the framebuffer after drawing.
3. Doublebuffering but did not swap buffers.
4. Geometry (vertices) as specified, translated or rotated is outside the viewport (i.e., clipped out).
5. Scaling to zero size.
6. Viewport is outside the window or has no height or width.
7. ...