Keyboard function in glut:
Set keyboard call-back to listen on keyboard: glutKeyboardFunc(keyboardFunctionName);
-Note that this is already in main in tutorial code (currently commented out)

Keyboard function: void keyboardFunctionName(unsigned char key, int x, int y):
    key - key pressed
    x - x-coordinate of mouse at key press
    y - y-coordinate of mouse at key press
    Can change name of function that processes keyboard input, but requires the above specific sequence of parameters

Reference for keyboard function:
https://www.opengl.org/documentation/specs/glut/spec3/node49.html

Mouse button press function in glut:
Set mouse button call-back to listen on keyboard: glutMouseFunc(mouseFunctionName);
-Note that this is already in main in tutorial code (currently commented out)

Mouse function: void mouseFunctionName(int button, int state, int x, int y):
    button - mouse button pressed (GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_RIGHT_BUTTON for left, middle, and right mouse buttons)
    state - state of mouse button press (GLUT_UP or GLUT_DOWN to indicate mouse button down and mouse button up)
    x - x-coordinate of mouse
    y - y-coordinate of mouse
    Can change name of function that processes mouse input, but requires the above specific sequence of parameters

Reference for mouse function:
https://www.opengl.org/documentation/specs/glut/spec3/node50.html

First, write functions that print output to console on keypress/mouse button press to know the functions work:
1. Add keyboard function to register input from keyboard
   a. Use keys ‘e’ and ‘r’ for input initially
   b. If keypress is ‘e’, print “KEY: E” to console
   c. If keypress is ‘r’, print “KEY: R” to console

2. Add mouse function to register mouse clicks
   a. If left mouse button down, print “MOUSE DOWN at (x, y)” to console where (x, y) are the current mouse coordinates
   b. If left mouse button up, print “MOUSE UP at (x, y)” to console where (x, y) are the current mouse coordinates

Next, add set of keyboard/mouse functions to start/stop animation

To be able to stop/start animation, add a global variable “int runAnimation” (0 - don’t run animation; 1 - run animation; could also use bool)

Switch myTimerFunc (given in tutorial 1 and used for animation) from:

```cpp
void myTimerFunc(int value)
{
    if (circleRadius >= 0.1)
        circleRadius -= 0.1f;
    else
        circleRadius = 1.0f;

    glutPostRedisplay();
    glutTimerFunc(1000, myTimerFunc, 1);
}
```

to

```cpp
void myTimerFunc(int value)
{
    if (runAnimation == 1)
    {
        if (circleRadius >= 0.1)
            circleRadius -= 0.1f;
        else
            circleRadius = 1.0f;
    }

    glutPostRedisplay();
    glutTimerFunc(1000, myTimerFunc, 1);
}
```
to control animation (now when runAmination != 1, circleRadius now doesn’t change w/ timer function so no animation...timer function continues to be called so animation can be re-started w/ change to runAmination value)

3. Add keyboard function to start/stop animation
   a. Start animation w/ ‘e’ key (adjust runAmination variable so animation runs)
   b. Stop animation w/ ‘r’ key (adjust runAmination variable so animation doesn’t run)

4. Add mouse function to start/stop animation
   a. Start animation w/ left mouse button click
   b. Stop animation w/ right mouse button click
   c. Then adjust to start and stop animation w/ left mouse button click (check runAnimation variable and change it to do “opposite of current...stop animation is running, start animation if stopped)

Then, add keyboard/mouse function to increase/decrease circle radius size

No longer doing animation; remove call to glutTimerFunc in main so timer function no longer called for animation

5. Add keyboard function to increase/decrease circle radius
   a. Increase radius w/ ‘e’ key press
   b. Decrease radius w/ ‘r’ key press
   c. Note that glutPostRedisplay() call needed at end of function to tell program to re-draw w/ adjusted radius

6. Add mouse function to increase/decrease circle radius
   a. Increase radius w/ left mouse button click
   b. Decrease radius w/ right mouse button click
   c. Note that glutPostRedisplay() call needed at end of function to tell program to re-draw w/ adjusted radius

Finally, add mouse function to draw square in location specified by mouse

7. Add mouse function to draw square in specified location
   a. Given click at location (x, y), draw square where location (x, y) is the lower left corner of the square
   b. Size of square should be the same regardless of location (to start make square side be of length 0.5 in OpenGL world that goes from -1 to 1)
   c. Need to convert coordinates from screen to the OpenGL "world"

   Screen from 0-256 in x and y directions where upper right is (256, 0) and lower right is (256, 256)
   OpenGL world from -1 - 1 in x and y directions where upper right is (1, 1) and lower right is (1, -1))

   Note that the y coordinates on the screen decrease when moving “up” on the screen but the y coordinates in the OpenGL world increase when moving “up”...need to take this into account when converting (x coordinate movement is the same in screen/OpenGL world)
Equation to convert (X, Y) on screen to world for this problem:

\[(x_{\text{world}}, y_{\text{world}}) = \left(\frac{(X_{\text{screen}} \times 2.0)}{\text{WIDTH}_{\text{SCREEN}}}, 1.0, \right.\]
\[
\left.1.0 - \frac{(Y_{\text{screen}} \times 2.0)}{\text{HEIGHT}_{\text{SCREEN}}}\right)\]

where WIDTH_SCREEN=256 and HEIGHT_SCREEN=256 in this problem

**To draw the square:**

i. Add global “float xStart” and “float yStart” variables so adjustments made in mouse function can be used in display function

ii. Set xStart and yStart in mouse function (read in mouse click, then convert from screen to world coordinates via above formula)

iii. See display1 function for example of drawing square, but note that the upper left coordinates is now (xStart, yStart) and square side is length 0.5

Coordinates of square are (xStart, yStart), (xStart + 0.5, yStart), (xStart + 0.5, yStart + 0.5), (xStart, yStart + 0.5) ...need to use this in display function

iv. Also glutPostRedisplay() is needed at end of mouse function to tell the program to re-draw the square at the specified location