Lab: Pair Programming with Graphics, Functions, and Local Variables

Pair programming is a technique where two programmers work at a single computer to write code. Person A is called the "driver," and is the person responsible for typing the code. Person B is called the "navigator," and is responsible for reviewing each line of code as it is typed in, looking for syntax errors, other bugs, or ways to improve the code to make it clearer, simpler, or more efficient. Before anything happens, both people should agree on the general structure of the algorithm they're going to use. The two people work together as equals, and switch roles frequently. The goal of this is to have one person (the driver) who focuses all their attention on the role of writing the code, while the other person (the navigator) focuses on making sure what the driver is writing is the best possible code they could write.

- Create a program to draw a bullseye (looks like the Target logo) on the screen.
    - Draw at least four concentric circles using the draw_filled_circle function.
    - Alternate between two colors to get a bullseye effect.

- Modify your program to add a function called "`draw_bullseye`" that takes two arguments: the x- and y-coordinates of the center of the bullseye. Inside of your main function, call your new bullseye function twice to draw the bullseye at two different locations on the screen.

  Your function definition line will look like:
  ```
  def draw_bullseye(x, y):
  ```

- Modify your program to add a third and fourth argument to your bullseye function called `color1` and `color2`. These arguments will let the caller of the bullseye function choose two alternating colors for the bullseye.

  For instance, if the user wanted a bullseye centered at (100, 100) colored red and black, they should be able to call the function like this: `draw_bullseye(100, 100, "red", "black")`

- Modify your program so the user can enter --- from the keyboard --- the (x, y) coordinates of the center of the bullseye, and the colors they want to paint it.
    - This will require four separate input statements: one for x, one for y, and two for the colors.
    - ***The input statements should not go inside the bullseye function; they should be inside the `main` function, and the information typed in should be passed as arguments to the bullseye function***.

- Modify your bullseye function definition to take a fifth argument, the radius of the bullseye. You'll need to do some math to figure out what the radii of the nested circles should be.

- **Challenge**: Write a function called `draw_square(x, y, side)` that takes as arguments the (x, y) coordinates of the center of a square and the length of a side. You should use `draw_line`, `draw_polyline`, or `draw_rect`. Remember: x and y should be the center of the square, not a corner.

- **Challenge**: Modify your program so the user can choose the center of the bullseye using a mouse click. Refer to the graphics library handout.

- **Challenge**: Modify your program so the user can choose the center and radius of the bullseye with two mouse clicks (first click chooses the center, second click chooses the a point on the border of the bullseye from which you can compute the radius). You'll need to break out your distance formula for this one.