CS 21 Programming Exercise: Week 7 Spring 2017
Due 9:00am Tuesday April 4th

We are going to program an old-school (8 bit!) Target Practice game. But asking the user to click in the center of the target is a bit too easy. Instead, they will have to 'bounce' a ball at our target.

After a greeting is drawn and removed from a graphics window, a three-part target will appear on the screen. Because we are going old-school, the target will be a square inside a square inside a square, all sharing the same center and all nicely colored and placed. The user will receive 2 points for getting their square 'ball' into the outer box, 5 points for the middle box, and 10 points for hitting the center! Each attempt starts with a spot shown on the screen outside of the target. This is where the player is 'standing'. The player chooses, via the mouse, a second spot, one that is to be one-half of the way between the initial spot and the intended final spot. When the ball is "thrown" it will bounce first on the chosen spot and then fly through the 'air', hopefully landing somewhere on the target. The user will have three tries, with all three remaining visible, after which the program will conclude with a final score and some level of congratulations.

Requirements & Notes:

1) All text will appear on the graphics window.

2) Pick a fairly large graphics window (900 by 900?) for the play area. Decide where you want your target, where you will place your directions and score. Place (0,0) in the bottom left, as we usually do. Pick the initial point (at random) in the bottom third of the screen.

3) Use functions to make your main as reasonably short as possible. No real computation should occur in main: it should be mostly just a loop and calls to functions. (At this point in the course we are advanced enough that there is lots of room for individuality, and so the actual number of steps in your loop will depend on some of the choices you make.) Think of **main()** as the manager – it doesn't actually do too much work, but spends its time gathering and distributing data from the places where the real work occurs.

4) A graphics window is a local but mutable variable. (Thus it is only visible to the function that creates it unless passed to another function, and any change to the copy of it passed to a function affects the original version.) I recommend creating it in main, and passing it to any function that needs access to it. Because it is mutable it doesn't need to be returned. On the other hand for the purposes of this assignment, all other graphics objects will be treated as local and immutable, and so *must* be passed both in and out of functions.

5) My suggestion is to do all drawing in main()

6) Functions must be well commented (what do they do, what comes in, what goes out)

7) Likely functions include:
   a. introduction() : Writes the introduction to the screen, pauses until user has read it (signified with a click) and then removes the text.
   b. makeSquare() : Creates an square of the desired size, radius and color, and sends it to the calling function (ie returns it).
   c. pickInitialPoint() : This will choose at random the point at which the user is 'standing'. Make sure it appears as a visible small square.
   d. pickBouncePoint() : Prompts the user to pick their bounce point, and returns it to main(). Make sure it is appears as a visible small square.
   e. findFinal() : Uses the initial point and the user's bounce point to compute the final landing point, which is returned to main()
   f. isInside() : Take a point, and square, and determines whether the point is inside the square. If the corners of a square are (x1,y1), (x2,y1), (x1, y2), (x2,y2), then a point (x,y) is inside the square if x1<= x<=x2 and y1<=y<=y2.
   g. computeScore() : Takes the location of the landing point and information about the targets and computes the number of points that turn earned
   h. result() : Prints a message based on the total number of points the user earned. A score of 9 or 12 might get a "9 points: Pretty good! Try again", while 26 might get "Almost Perfect!" and 30 "You've earned the Big Gorilla!"
   i. remove() : Once you have created an object, append it to a list you are keeping in main of all the objects that have been drawn. After the game is complete, send that list to this function that will undraw those things to clear the screen.

8) If (x0, y0) is the 'center' of a square, then the corner points are at (x0 +- d, y0 +-d), where d is a constant. You can choose the color and comparative sized sof the squares.
9) I will expect a function when (1) Some bit of code is re-used, and (2) Some bit of code exists primarily to produce some result.