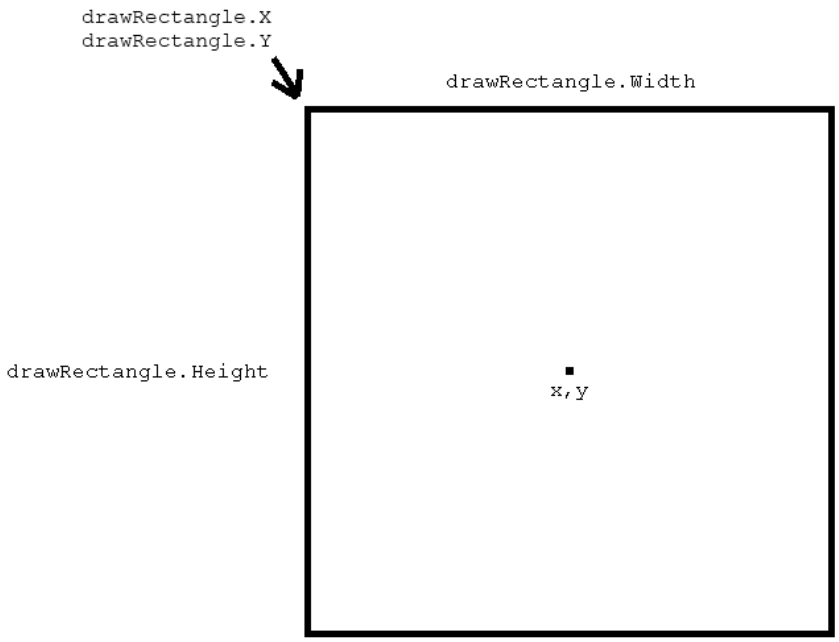
**Programming Assignment 4**
**Step by Step Instructions**

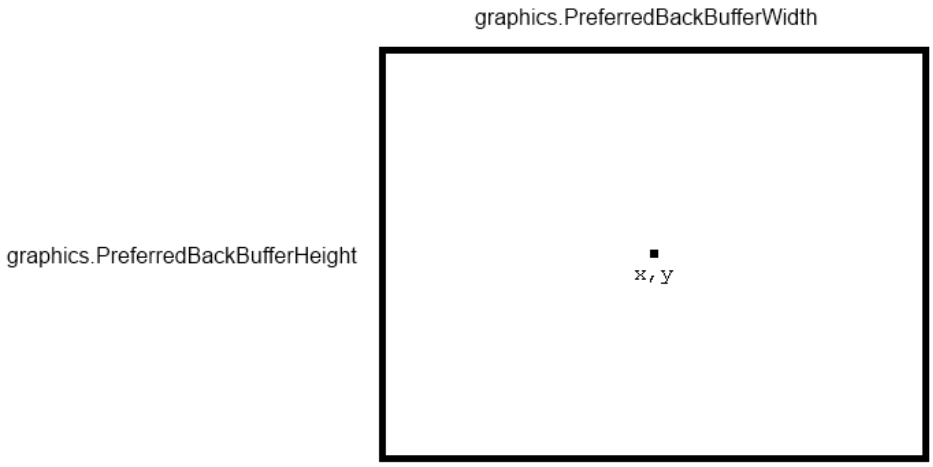
Start up the IDE on the ProgrammingAssignment4 project you extracted from the zip file you downloaded. You must use this project as your starting point.

Throughout the code, I've sprinkled comments that start as // STUDENTS:. Those are all the places you need to add code. Although you could try to add all your code at once and hope it all works, that would be a REALLY BAD IDEA! Instead, I recommend that you complete the assignment using the following steps:

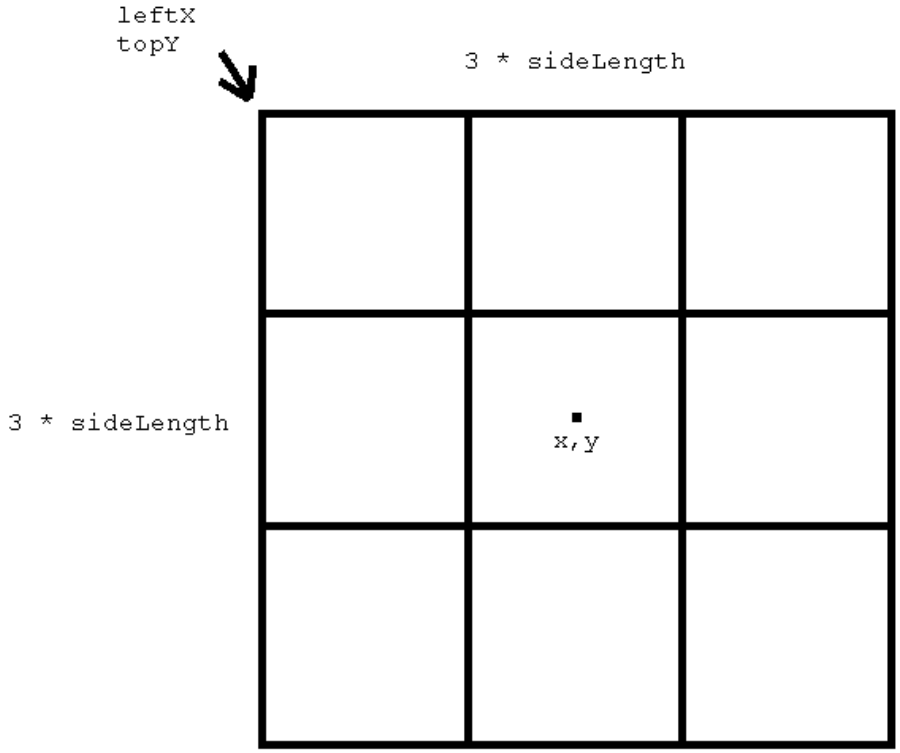
1. Run the project as provided to make sure it compiles and runs for you
2. Declare a variable to hold a `Board` object at the top of the `Game1` class
3. Create the `Board` object in the `Game1` `LoadContent` method. Just pass in 0 for x and 0 for y for the center of the board at this point (we'll fix that later)
4. Draw the board in the `Game1` `Draw` method
5. Draw the square sprite in the `BoardSquare` `Draw` method. Note that this is drawing a `Texture2D` like we discussed in class and Section 5.3 of the book. Look at the fields in the `BoardSquare` class to find the appropriate rectangle to use here
6. Run the code. You should see (what appears to be) a single board square in the upper left corner of the window. That's actually 9 board squares drawn on top of each other
7. Change the code in the `BoardSquare` constructor to locate the square properly. A picture really helps here as you try to figure out the equations:




   At this point in the code, x and y have been passed in as parameters and `drawRectangle.Width` and `drawRectangle.Height` have already been set properly in the `LoadContent` method. Your job in this step is to calculate `drawRectangle.X` and `drawRectangle.Y` given those values.
8. Run the code. You should see all 9 squares (some may be cut off by the edges of the window), but the board still isn't centered in the window
9. Change the code you added in Step 3 to pass in an x and y that will center the board in the window. You won't see any change until after you complete Step 10, but we need to make these changes in two steps. Again, a picture may help:




10. Change the code in the `Board` constructor to locate the board properly. You'll probably find the static `BoardSquare` `GetSideLength` method helpful here. Assuming you call the `GetSideLength` method and put the returned value in a `sideLength` variable, here's yet another helpful picture:




11. Run the code. You should now see the entire board centered in the window
12. Add the code in the `BoardSquare` `Draw` method to draw the x sprite if the square contents are equal to `SquareContents.X` or draw the o sprite if the square contents are equal to `SquareContents.O`
13. Uncomment the code after the first comment in the `BoardSquare` `Update` method and fill in the required Boolean expression and code in the if block
14. Delete the code at the end of the `BoardSquare` `Update` method as instructed by the comment
15. Add the code after the first comment in the `Game1` `Update` method to let the player take a turn. Note that the method returns a `bool` that you need to save to use later.

16. Run the code. You should be able to click on the board squares and see an X appear in each square you click. Of course, the computer never gets to take a turn – Cheater!
17. Uncomment the code after the second comment in the `Game1` `Update` method to change whose turn it is if the player took a turn. This is where you'll use that `bool` you saved above.
18. Run the code. You can now only click one board square to put an X there
19. Add the code after the third comment in the `Game1` `Update` method to have the computer take a turn and change whose turn it is. The computer variable is a `ComputerPlayer` object that you can tell to take a turn. To change whose turn it is, look at the code from Step 17
20. Run your code using F5. You should now be alternating turns with the computer. Note: If you play until the board is full, you'll need to forcibly shut down the game. Go to the IDE and select Debug -> Stop Debugging
21. Declare a variable to hold a `QuitButton` object at the top of the `Game1` class
22. Create the `QuitButton` object in the `Game1` `LoadContent` method. You want the game state to change to `GameState.Quit` when the button is pressed, so you should pass `GameState.Quit` as the last argument to the constructor
23. Add the code after the last comment in the `Game1` `Update` method. If you're wondering how you can tell that the game is over, you should realize that the board can tell you whether or not the game is over if you ask it!
24. Add the code to the `Game1` `Draw` method to draw the quit button if `gameState` is equal to `GameState.GameOver`
25. Run the code. When the game is over, the quit button should appear (though you can't do anything with it yet)
26. Add the code after the fourth comment in the `Game1` `Update` method to update the quit button
27. Run the code. Everything should work now, with the quit button letting you quit the game
28. Go drink a (legal) beverage of your choice