Practice using good techniques to write the programs below. Try to write efficient code. Use comments as needed to explain your code.

A. Tic-tac-toe (25 points)

In the Tic-Tac-Toe example done in class, we saw that there was a major problem with the function for entering player's choice of box — the player could enter anything, including numbers for boxes that had already been chosen. Of course, this is quite lame.

Fix the function so that program checks to make sure that the choice entered is valid (an integer between 0 and 8) and that has not already been chosen. If a player tries to enter an invalid choice, the program a warning and give the user another chance to enter a proper value.

B. Struct 1 (25 points)

As a simple first exercise with structs, we will create a program that adds two distances, where the distances are defined in terms of feet and inches.

1. Define the template for a struct with the tag of distance. The struct has two members, an integer called feet and a double called inches.
2. Declare three variables using the struct template: `d1`, `d2`, and sumOfDistances. You can do this as part of the struct definition or do it within main.
3. Within main, use `scanf()` to enter values for the feet and inches of both `d1` and `d2`.
4. Then calculate the sum of `d1` and `d2` (feet and inches). Note if inches ends up being bigger than 12, the value of feet should be adjusted appropriately. For example, if the sum ended up as 5 feet 18 inches, it should be re-expressed as 6 feet 6 inches.
5. Print out the resulting sum.

C. Struct 2 (25 points)

We have seen that a struct can contain all the familiar variables – char, int, double. It can also contain arrays, strings (which are arrays of characters), and pointers.

This second exercise will focus on using arrays as members within a struct. Our theme for a program to see how this works is a mini set of medical records.

1. Start by defining a struct template that has the following members:
   a. a patient ID number, which is some integer between 1 and 100000.
   b. a patient name in the form of a character array — just a single first name is adequate.
   c. a three-element array of integer for the month, day, and year (eg. 4, 2, 2018).
   d. a double for body temperature,
   e. a integer for weight (in pounds),
   f. a integer for height (in inches),
   g. a two-element array of integers for blood pressure (110, 70).
   h. a double for body-mass index.
2. Once the struct is defined, declare an instance of the struct variable.
3. Fill items *a - g* with reasonable numbers for a "typical" human being. You can simply assign values to the various members — this probably the easiest approach — or you can use scanf() to have the user enter the values. Either way will be will be slightly tedious.Note that you will probably want to use strcpy() to initialize the name string.
4. Then calculate the BMI (body-mass index) from the height and weight and assign the value to the bmi member.
5. Finally, print our the patient's records.

Note, you can certainly implement some of these actions with functions, although functions are not a requirement.

D. Quiz (25 points)

As usual, there will be a short quiz. It should come as no surprise that the quiz will be on structs.