CSE 5462: Lab3 (100 points)
TicTacToe

Demo in Class or by appointment: February 11
Electronic Code Submit Deadline: 9pm, February 11

Important: You can work alone for this lab, but you must each turn in your OWN code. Do not write one version and submit it twice. That will constitute academic misconduct. Please identify who you work with in your code.

We will demo in class or in lab. For in class demos, we will go around the room. Be prepared to demo at the start of class. The submitted code will be used only to verify that you did not copy from others, to compile and re-run your program, to make sure you were indeed demonstrating your own code, and to grade for documentation of your code. You may be asked to connect your client/server to a client/server from another pair of students!

- For socket programming use can either use the CSE linux system (csegrid) or your own machines. For your own machines, demo can be done in classroom.

In this lab, you are given a file called tictactoeOriginal.c. That code allows 2 players on a single machine to pass-and-play a game of tic-tac-toe. The purpose of this lab is to modify that program so that it can be played by 2 people, running on different machines. You will use STREAM sockets. You will use a predefined format for the data (see below).
You may write this as a single application or as a client/server pair. The command line input will be:

- portnumber that the programs will communicate on,
- player number (1 or 2), where player 1 is the player creating and binding to the socket
- ip address of player 1 (this will only be used by player 2)

This code is the basis for lab 4, 5, etc.

Submit well-documented and well indented code along with a README file explaining how to run the program. You also need to submit a makefile. Submit it using GitHub, in a subdirectory called Lab3

The initial format (think protocol) of what is sent between client/server:

4 bytes – representing the square that player is moving to

The grading rubric is as follows:

1. Program correctness and robustness: 70% – Application has to work to get these points
2. Coding style (e.g., comments, indentations): 20%
3. Documentation (the README file and design document): 10%

What you should get from this lab?
Tools:

- basic connection oriented socket IO (socket(), bind(), listen(), connect(),close(),read(), write() etc);
- bad return codes from read()
- arrays/structs in C;

Concepts:

- how to modify/re-purpose someone else's code;
- how to 'net enable' an application written for a single machine;
- determining when the other application has hung-up on you;
- how to gracefully handle errors;
- how to handle garbage sent (intentionally or not);
- how do you understand and debug code you didn't write;