Objective

In this project you will complete a program that plays Tic-Tac-Toe with other computers on the network. In addition, you will gain experience with adding functionality to an existing program that was written by someone else.

Preparation

Since your program is supposed to play against someone else's program, it is important that everyone's code agree on the necessary communication parameters and commands for the game. The header file, `TicTacToe.h`, defines constants that everyone should use. Two files, `getServers.cpp` and `playTicTacToe.cpp`, have been provided that need some code inserted (see the lines tagged with '`/****`'). In addition, your professor has written several other files, `TicTacToe.cpp`, `clientMain.cpp`, and `serverMain.cpp` that need ***not*** be modified.

You will also need to download the following utility files from EASEL: `connectsock.cpp`, `passivesock.cpp`, `UDP_recv.cpp`, `UDP_send.cpp`, `timestamp.cpp` & `wait.cpp`. (The C++ `main` function is located in `TicTacToe.cpp`.)

Tasks

1. In the `playTicTacToe()` function (found in `playTicTacToe.cpp`), add code to implement the comment:

   ```
   // Send move to opponent
   ```

   The code prior to this comment asks the user to enter a move for the game. (Moves are digits from 1 to 9).

2. Add code to the `playTicTacToe()` function that (1) receives the opponent's move via the network, (2) calls a function that will update the game board to reflect the opponent's move, and (3) call a function that will display the updated board.

3. Add code in `getServers.cpp` that will send the `TicTacToe_QUERY` message to your LAN's broadcast address. Be sure to use the `TicTacToe_UDPPORT` port number.

4. Add code in `getServers.cpp` that will (a) receive the responses to the broadcast message and (b) collect the information in an array of `structs`, named `server[]` (see `TicTacToe.h` for a description of the `struct`).

Hints

1. The C++ string objects are nice to work with; however, the `winsock2.h` functions all work with C strings (`char` arrays). Consequently, you will either need to convert to & from C++ and C-strings, or use C-strings exclusively. You may find the following C string functions helpful: `strlen()`, `strcpy_s()`, `strcat_s()`, `stricmp()`, `strncmp()`, `strstr()`.
2. You may also find it necessary to convert from an integer to a string or vice versa. The function `atoi()` converts a C-string to an integer. If the string doesn't contain an integer, the function returns 0.

   The function `_itoa_s()` converts an integer to a C-string.