COP4521 Programming Assignment 4: Online Tic-Tac-Toe Game
(Due Oct. 30, 2024, 11:59pm)

Objectives:

- Developing a networked application with socket programming

Description:

You can do this assignment either by yourself or in a group of 2 people. In this assignment, you will develop an online Tic-Tac-Toe game application that allows two people to play the Tic-Tac-Toe game over the Internet. The application consists of two programs, a tic-tac-toe server and a tic-tac-toe client. When playing the game, the person who is the second to make a move should start the server program on a machine with a selected port as a command line argument. Then, the person who will move first will start a client to connect to the server (server machine name and port number as command line arguments), and make the first move. After that, the two players will take turns to make moves until the game is over. The game finishes when one of the players wins the game or when the board is filled and no more move is possible; in which case, the game is a tie. When a game finishes, the app reports the results to both players. After that, the server loops back to wait for the client to start another game while the client exits (the player can restart the client to play the next game). An example game trace is shown below:

Server (second player):

```
<linprog5:532> python3 server.py 55000
server.py 55000
Waiting for opponent to connect....
Receive opponent connection from  ('128.186.120.190', 33030)
   1 2 3
A  . . .
B  . . .
C  . . .
Waiting for opponent's first move. Don't type anything!
   1 2 3
A  # . .
B  . . .
C  . . .
Your opponent played A1, your move([ABC][123]): B2
   1 2 3
A  # . .
B  . O .
C  . . .
Wait for your opponent move (don't type anything)!
   1 2 3
A  # # .
B  . O .
C  . . .
```

```
Your opponent played A2, your move([ABC][123]): B1
   1 2 3
A  # # .
B  O O .
C  . . .
Wait for your opponent move (don't type anything)!
   1 2 3
A  # # #
B  O O .
C  . . .
Your opponent won the game!
Waiting for opponent to connect....
```

Client (first player):

```
<linprog6:531> python3 client.py linprog5 55000
   1 2 3
A  . . .
B  . . .
C  . . .
Enter a move([ABC][123]): A1
   1 2 3
A  # . .
B  . . .
C  . . .
Wait for your opponent move (don't type anything)!
   1 2 3
A  # . .
B  . O .
C  . . .
Your opponent played B2, your move([ABC][123]): A2
   1 2 3
A  # # .
B  . O .
C  . . .
Wait for your opponent move (don't type anything)!
   1 2 3
A  # # .
B  O O .
C  . . .
Your opponent played B1, your move([ABC][123]): A3
   1 2 3
A  # # #
```

```
B  O O .
C  . . .
Congratulations, you won!
<linprog6:532>
```

Due time: October 30, 2024, 11:59pm.

Submission: Name your Tic-Tac-Toe game server and client lastname_firstinitial_tictactoeserver.py and lastname_firstinitial_tictactoeclient.py, respectively. Put the two programs as well as any other supporting files in a tar file and submit through canvas.

Grading (70 points total):

- Include basic header (template at the course website) for assignment and name your programs lastname_firstinitial_tictactoeserver.py and lastname_firstinitial_tictactoeclient.py. *List team members for the assignment in the basic header*. (10 points)
- Successfully start the server on a port as the command line argument (5 points)
- Successfully establish the connection between the client and the server; servers reports the client's IP address and port number (5 points)
- Successfully display the initial board on both client and server (5 points)
- First player (client) successfully makes the first move; the move is displayed on both the client and server (10 points)
- Second player (server) successfully makes the first move; the move is displayed on both the client and the server (5 points)
- Server and client report wrong moves when they happen (move outside the board, play on a position that has already been occupied, etc) (5 points)
- Successfully play to the end of the game with a winner and declare the winner to both the client and the server (10 points)
- Successfully play to the end of the game that is a tie and declare the result to both the client and the server (5 points)
- Server loops back to wait for another connection while client exits after a game is finish (5 points)
- Server starts the second game correctly (5 points).
- You can assume that the players take turns to the end of the game nicely. If other situations such as players making moves out of order or client/server crashing/losing connection in the middle of the game, the program's behavior can be non-deterministic.

Notes:

- A reference implementation has about 150 lines of code for each of the client and the server. The code size can be smaller if the game engine is in a separate module.