Protocol for Lab 6- CSE 5462 - Spring 2021
Version 4

Entire datagram to be sent across(listed in order): **Max size of Datagram = 40 bytes**

1. One byte for version # (In this lab it is 4, 0x 04)
2. One digit byte for sequenceNumber
    a. Client starts seqNumber at 0
    b. Each side increments it by 1 on successful receipt from peer
        i. So client sends seqNum=0 for its new game request
        ii. Server sends seqNum = 1 back for its first move
        iii. Client sends seqNum=2 for its first/next move
        iv. and etc
3. One digit byte for type of command
    a. Hex 00 = new game
    b. Hex 01 = move
    c. Hex 02 = game over
        i. Client is sending the last move. Its move wins the game. Client waits for a GAME OVER command from server to verify that the server got its winning move
        ii. This works in reverse too; if server sends winning move, it will wait for GAME OVER command from client to verify that the client got its winning move
        iii. So you wait for a GAME OVER command to confirm that the other side received your winning move
        iv. If you send GAME OVER you need to stay open for the grace period (60s) before you close
4. One ASCII character for position (assuming command is move)
    a. So move 1 sends byte 0x31
    b. Move 2 is byte 0x32
    c. etc
5. One digit byte for gameNumber - assigned by server when game created

Datagram Format

Protocol Version	Sequence #	Command	Position	Game Number			

Example Datagrams

- EX: 0x 04 01 01 31 01 = Version 4, sequence number 1, command is move, moving on square 1 (moves are ASCII characters), game #1,
- EX: 0x 04 00 00 = Version 4, sequence number 0, command is new game
- EX: 0x 04 0A 02 00 04 = Version 4, Sequence # 10, Game over command, no move, Game Number 4

Tic tac Toe grid layout

1	2	3
4	5	6
7	8	9

Player 1 is the "server"

- The server / player 1 goes first
    - Responds to client new game request with the first move.

Player 2 is the "client"

Other Specifications:

- **If server has a game running and gets another new game request, it handles it!**
- **Timeout Time: 30s**
- **On any errors (that aren't otherwise specified), close the socket**
- **Using Datagram sockets**
- **Server can handle multiple games….server MUST BE single threaded.**
- **Server will ignore new game requests if already full**
- **Server will ignore moves if the game doesn’t exist**
- **Clients are allowed to exit if they receive bad data**
- **If receives GAMEOVER command, ignore the move field**
- **If we receive a duplicate datagram that is one sequence number back (I sent 4, but you send back 3) Then resend 4. Duplicates that are more than one back are ignored.**
- **Datagrams that are ahead of sequence number (i send 4 you send back 6+) then left up to implementer**

SOME SCENARIOS

- If the client asks server for a game, but the server can't run any more games…
    - Server just doesn't reply (so client will timeout on their end)

- If the client sends a move to the server for a game that no longer exists…
    - Server just doesn't reply

- If the client gets data that it was not expecting …
    - It can ignore it and loop back to a recvfrom(); or it may exit

Example game with what occurs with sequence #:

Client: Send 0 (new game)
Server: Receives 0, initializes game, sends move in datagram with sequence # 1
Client: sends move back in sequence #2
Server: Sends move back in sequence #3
...
Client: Wins game, sends move command with winning move and next seq number
Server: sends back game over command to client
Client: ends game

<u>QUESTIONS</u>

New name for "move"?

Result: position

Sequence Number vs Move Number (naming)?

Result: Sequence Number

Client sends 0 seq num to server. Does server send back 1? If server sends 1, does client send 2?

Result: Client sends seq 0 when starting the game. Server will send back 1. Client send back 2

Which byte is the sequence number?

Result: second byte

<u>END OF QUESTIONS</u>