I used regular pthread mutex that i global defined as a lock just like it was in the recitation. In my function that I send threads into, i take a parameter which is the player specification that can be either 'x' or 'o'. I used pthread_create function and sent the parameter with it. In order for one player to do the necessary checks, I lock the mutex so that while a thread is checking the table (using isFull() and checking if the matrix coordinate to be filled is filled or checking if the game ended), the other thread can not change it. Similarly, A player can not overwrite the other player as we lock before doing anything to table. So it is safe.

I used winner as a global variable to track the last player that changed the matrix. After the function ends, the last player is winner if its not a tie. The current player can not be the last player, so the players go in rounds (1 by 1, in order)

My thread function as a pseudocode:

Take a void pointer parameter

Shift parameter into char and store it player(x or o)

While game is not ended

Lock the mutex so that while we are doing certain processes, the other player wont get affected or doesnt affect

if winner(last player) is not the current player and game is not ended and matrix isnt full and realwinner isnt defined

Else

Create random coordinates

While random coordinates in matrix are filled,

create new coordinates

assign value to matrix

print the player and where they put the x or o into.

Change winner(last player) to current player

İf there is a winner or matrix is full

Bool changes to false so the game ends

Unlock the mutex

Return NULL to stop void function

After all, if the mutex doesnt unlock, we unlock at the end of the while loop. Here, we have no chance to unlock it twice since all previous unlocks we had also returns (stops) the function.