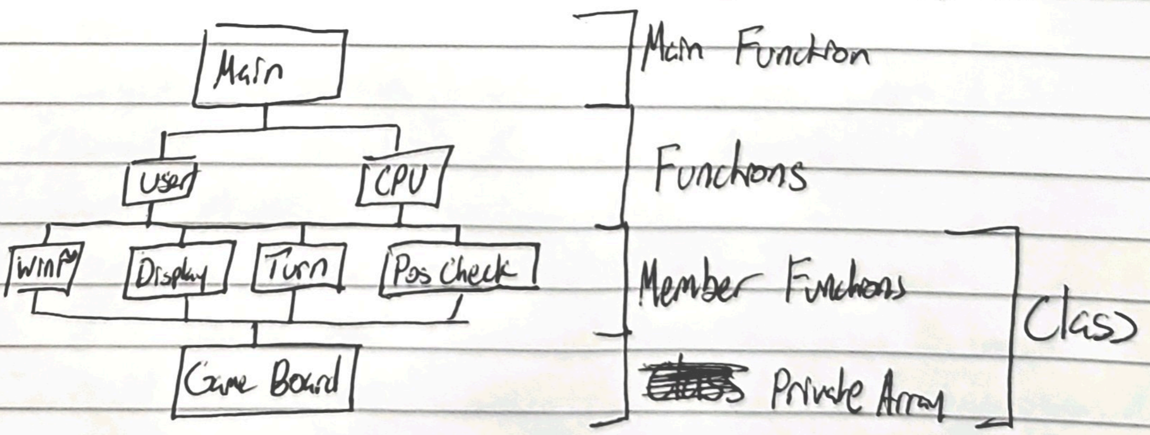
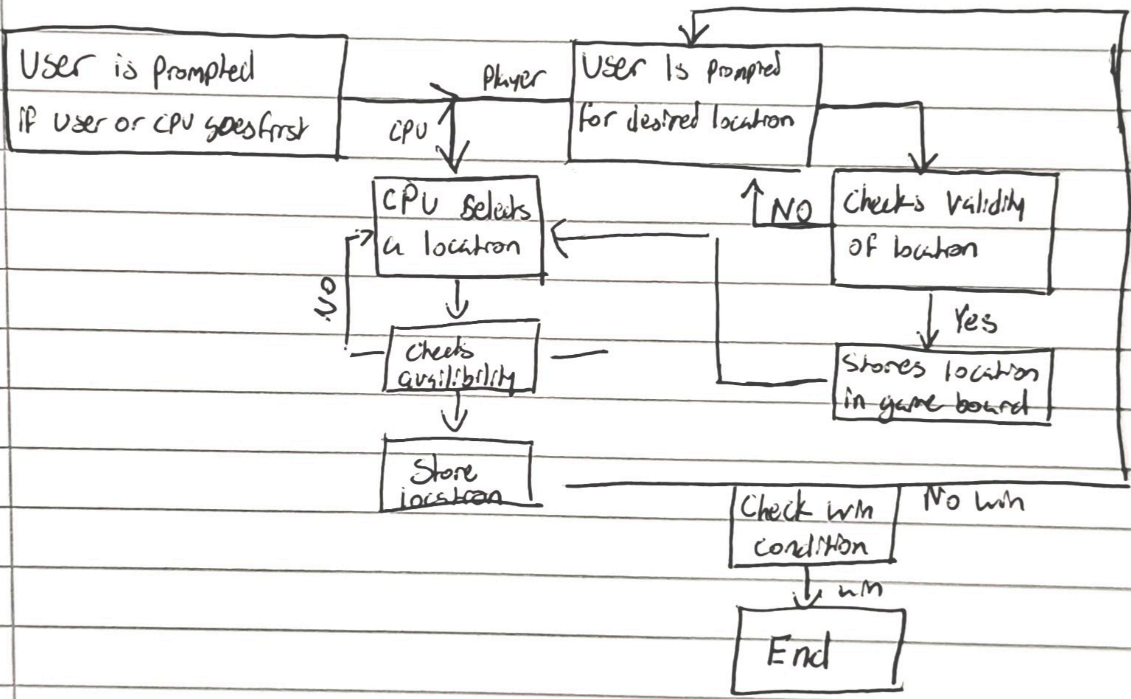
Tic Tac Toe Program

7-31 1:30
- Class structure to control game board, and player interactions. This [illegible] should make it simple to diagnose what issues I could have interacting with the board, such as missed checks, incorrect imputs, or errors displaying the board.
- Due to the class structure I should be able to [illegible] create member functions that modularly interact with the game board. This could include but not limited to, checking for the win condition, displaying the game board to the player, providing a medium for the player to change the game board, and possibly providing board information to the CPU player.
- If I have time I could implement a CPU that responds to the player, but I will start by building one with the rand / srand functions. Also due to the class nature of the board, I could later generate multiple games at the same time, or make it 2 player
- Due to the multiple conditions to end the game I could have a public member function interact with multiple private member functions that each are in charge of one win condition.
- I also should create a member function that checks the validity of a selected position when it recieves the users desired position

- Using the classes to interact with the board, I can have normal functions interface with its member functions as a tiered system



- Initially for the game board I was thinking of using a coordinate system, but that would be complicated for the user, so I decided to utilize a numbering system 1|2|3 4|5|6 7|8|9. The user would only need to input the number corresponding to the space desired. This also simplifies the board from a 2D array to a simple array. (Though the index is offset by one, because it is more natural for a user to start at 1 rather than 0)



User is prompted if user or CPU goes first
Player
CPU
User is prompted for desired location
CPU selects a location
NO
Checks validity of location
Yes
NO
Checks availability
Stores location in game board
Store location
Check win condition
No win
win
End

2:10pm 7-31 • Started coding, beginning with creating the class, and its member functions, with the intention of creating the functions that need to interact with the private array that acts as the game board.

- Created a member function that changes what character is placed on the board depending on the current player
- Created a member function that displays the current game board and clears all prior info from the console
- Created a member function that checks if input location still has its default value, if it does it replaces it with the current player's symbol

3:20pm 7-31 • Player function is created, allowing me to confirm the effectiveness of the previously made member functions. With the creation of the function I added a warning message to warn the user of improper inputs
• With the ability for me to manipulate the array from within the program now, I began working on the checks on win condition, I created 4 member functions for the different win conditions, matching 3 diagonally, horizontally, vertically, and running out of spaces to use. I then created a function within the class that systematically checks each functions and returns whether the win conditions have been met, and also displays an appropriate message depending on the condition met.

4:00pm • I then created the CPU, due to time constraints I was restricted to use randomly selected positions with the CPU with rand and srand. With the other player I can test the win conditions.
• The column and row checks failed to stop the program, and figured out that there were some logical errors that stoped the checks from propperly proceeding, and by adjusting them slightly the problem was fixed.

4:45pm • Program Completed following adjustments made to the main function to remove all debug code.
- Following completing the code, I left for work

2:30-3:00 - set up github and generated a repository as per instructions

<u>Project</u> 1:30 - 4:45 3hr 15min