- Something you did that wasn't covered in the videos
    - I coded my connect four ai to not only look for winning moves and block winning moves but also to think a move ahead by trying to create positions where it has two possible winning moves next turn so the other player cannot block it. I also did the opposite by trying to stop the other player from making those types of moves. When it can't find a move that does one of the above things, it just makes the move that creates the most three in a row for itself. (I copied the code from the hasWinner method)
- A problem you ran into and how you were able to solve it
    - A kind of problem I had was deciding whether to code the bot to play against real people or against the random CPU. When playing against real people, it is usually better to have the ai make a random move when it can't find one that is beneficial, but against the random CPU, it has a better success rate just going in the first column whenever it can't find a move as there is a good chance the random bot won't block it. I fixed the problem by just adding a line of code that I either keep or comment out depending on whether I'm testing it against the random CPU or having a real person play against it.
- A feature/improvement you'd like to add in the future
    - I would like to add some sort of recursive feature to it as my current ai can only think 1 move ahead. I think it would be interesting to allow the user to select the difficulty of the ai with a higher difficulty meaning the program would recurse more and therefore think more moves ahead.

Data (going second):

Against Random CPU (going first column when it can't find any beneficial move): 99.7%

Against Random CPU (going randomly when it can't find any beneficial move): 99.0%

Against simple AI (only wins or blocks wins): 86%

Against Real People: ~70%??

(going first)

99.9%

99.5%

90%

No data