Project #: 5
Semester: Spring 2022________
Name: Chris LaFave

I. Requirements: Restate the problem specification and any detailed requirements in your own words.
Recreate a Tic-Tac-Toe program using the GUI functionalities of wxWidgets.

II. Design: How did you attack the problem? What choices did you make in your design, and why? Show class diagrams for more complex designs.
At first, I just followed all the videos, and then when I finished those, I went down the rubric and solved those problems one at a time. I think my best choice was to make a 2d array of the gameboard to make it easier to check if the game was over. I also like my use of functions; I think they were all very useful.

III. Security Analysis: State the potential security vulnerabilities of your design. How could these vulnerabilities be exploited by an adversary? What would be the impact if the vulnerability was exploited?
I noticed that using breakpoints would bug the program into never moving on from X's turn. If somebody could lag the program enough it might let them have infinite turns.

IV. Implementation: Outline any interesting implementation details in your solution.
For some reason, saying tester[] == [Array of just X's or O's] in the CheckForWinner() function was not working as I thought it was supposed to. My original idea with that was to make an array with all O's and all X's (oWins[] and xWins[] respectively) and compare my tester[] array to that. But because that wasn't working, I realized that I could just check for two things: that every element in tester[] is the same, and that they aren't empty. If both of those conditions are true, then somebody won, so I can just return one of the elements in tester.

V. Testing: Explain how you tested your program, enumerating the tests if possible. Explain why your test set was sufficient to believe that the software is working properly, i.e., what were the range of errors for which you were testing.
At first, I had both the example TTT up and my program, but after I got far enough with my program, I only had mine open. I would attempt to implement each area of the rubric one at a time, and I would check with the debugger if what I just implemented worked properly.

VI. Summary/Conclusion: Present your results. Did it work properly? Are there any limitations? NOTE: If it is an analysis-type project, this section may be significantly longer than for a simple implementation-type project.
It works perfectly! I'm fairly sure that it works exactly like the example program, with minor graphical stuff that I didn't work out how to change, and the fact that it doesn't switch who starts each round.

VII. Lessons Learned: List any lessons learned. For example, what might you have done differently if you were going to solve this problem again?
I definitely would have brushed up on arrays more and not procrastinated as much as I did, but other than that it went very well! Most of the stuff I implemented worked first or second try, and I didn’t have to debug anything for longer than like 20 minutes.