One obstacle I overcame was VSCode having linkage issues. I had the Microsoft version (whichever has the blue logo that looks more like an α instead of an infinity) and that had several issues running anything. Initially it was somewhat of an issue of skill, since I stupidly wasn’t running the main file, but once I got over that a bunch of things weren’t linked. I’m still not really sure what issue I was having, but I figured out that there was another version of VScode, Visual Studio Community. That made a lot of sense. I reinstalled and had no issues. I think it was because my computer got wiped when I fried the motherboard from a battery replacing mishap.

Another obstacle was figuring out why nothing showed on the board. I couldn’t get set to work, evaluate just wasn’t useful, and the board didn’t even show up on initialization. The main suspect was the method that checked if it was OK to add, so I checked isValueOK(). Nothing there was wrong, and I even drew out a truth table, so I went to the next likely culprit, notFound. The logic there was exactly inverted, so I just put everything in parenthesis and added a not in the front.

Test data:

1. just start the program
   1. very good blanket test to see if anything’s wrong with adding numbers, because the pathway with which the board is initialized is the same after you get the values into the program
2. start the program with an almost correct board (one missing) and put the missing value in
   1. checks for functioning complete conditions
3. start the program with an almost correct board (one missing) and run evaluate, check if it’s 80, then put in a duplicate and see if it goes down
   1. checks fringe condition of an almost complete board
4. start the program and put a wrong number somewhere that is a duplicate in the box and not the row or column
   1. checks box OK condition
   2. also while we’re here, u run evaluate and see if the number went down 1 (it should go down 1)
5. start the program and put a wrong number somewhere that is a duplicate in the row and not the box or column
   1. checks row OK condition
   2. also while we’re here, u run evaluate and see if the number went down 1 (it should go down 1)
6. start the program and put a wrong number somewhere that is a duplicate in the column and not the row or box
   1. checks column OK condition
   2. also while we’re here, u run evaluate and see if the number went down 1 (it should go down 1)
7. start the program and put a wrong number somewhere that is a duplicate in the box and the row but not the column
   1. checks OK condition of multiple faults
   2. also while we’re here, u run evaluate and see if the number went down 2 (it should go down 2)
8. start the program and put a wrong number somewhere that is a duplicate in the box and the row and the column
   1. checks OK condition of every possible fault
   2. also while we’re here, u run evaluate and see if the number went down 3 (it should go down 3)