Understanding

- I need to create a program called symArrays
  - It should check to see that arrays are symmetrical
  - It should take three integers on the command line
  - It should check to make sure enough integers were entered for the array and allow the user to re-enter
  - Array's need to be dynamically allocated
- I need to create a program called sameSum
  - This must check the rows, cols, and diagonals of a 2-d array to make sure they all match
  - Max 10 rows/cols
- I need to create a ticTacToe program
  - It should allow command line input for number of games
    - but play 1 game if no command line input
  - Print out the board
  - Tell what user is moving and take input
  - Play as many games as requested
  - Display overall winner and scores

Design - ticTacToe

```
gameloop
{
        gamecount = 0
        board;
        firstmoveplayer = X
        playerwithmove = X

        loop
        playerwon = false
        moves = 0
        playertomove = playerwithfirstmove
        switchplayerwithfirstmove()
        print new game and initialize
                loop
                        print board
                        get input
                        validate input
                        move made?
                        win?
                        tie?
                endloop
        add to win total and game count
        if game total is done, exit to main
}

main
{
```

```
        parse for valid input
        gameloop(num games and score references)
        print scores and calculate overall winner
}
```

Testing

Input	expected output	output
./ticTacToe 4	four games played	same
./ticTacToe	one game played	same
0 0 // X moves 0 0 //invalid! 0 1 //O tries again 1 1 //X moves 0 2 // O is kinda stupid…. 2 2 //x wins	O gave invalid input and should be given a chance to play again. X wins. O moves first next time if there is another game. If not, X is overall winner. Total displayed.	Same

Reflection

I think that most of my challenges were in things that I assumed, but didn't hold true. For example, I made the assumption that I could only have one variable, keeping track of who moved last. This wasn't the case though, because it isn't true that the last player to move will not be the first player to start the next game. If we want to have true alternating games, I need a variable to track the current move state, and a variable to track who will start the next game. I think that this shows that a programmers job transcends just creating the right technical decision. There are design decisions and business assumptions that will be made and could potentially be devastating to a project. It was only after I stepped back and checked my assumptions that I was able to figure out why the start player wasn't alternating.

I also think that this is giving us good experience in never trusting the user for correct input. In a perfect world, this would happen. However, our programs should not crash and burn just because a user makes an errant keystroke. This is good experience for me, as my job mostly deals with creating connecter programs that other machines use. I don't have a lot of experience with UI design, so this is getting me thinking about that.