1. After completing the Voting Poll contract user manual, simulate an election with 3 candidates. Give each of the candidates one vote, and make sure that the functions are returning the correct values. What two values do you have to add into the "addCandidate" section? Implement a function called "getColor". to also store a Candidate's eye color as a string variable and display it.

2. After completing the Auction contract user manual, simulate an auction. Remember that you have to change the value of the "msg.value" variable from zero, before you can call the "makeBid" function. Simulate an auction for any item you choose. Make the first bid with "900 gwei", the second bid with "29000000000000 we"i, and the last bid with "5 ether", who won the auction?

3. After completing the CRUD contract user manual, create a simulation of storing users in an array, reading the user's name from the address and return the name. How would you implement a function that will store an amount of currency for the user into a wallet? Simulate this new function by returning the user's wallet value.

4. Msg.sender and msg.value are used frequently in the creation of these smart contracts, what does msg.sender and msg.value refer to? Write function(s) that will return the value of "msg.value" and "msg.sender" of a certain "user" struct.

5. After completing these smart contract user manuals, you should have a general idea of how smart contracts are made. Write the smart contract logic for a betting game. The betting game will have a "better" struct and a "game" struct. The "better" struct will have the info for the person placing the bet, and the "game" struct will have the variable for the game's logic. Create logic to simulate a betting game, where the better stores a bet (as an integer variable) and this bet has to be higher than the minimum bet. The minimum bet should be initialized in the constructor of the contract.