Assignment1

	URL on Etherscan
Venus account	https://goerli.etherscan.io/address/0x220fe5c8bb71ad05aa659055623a84d7168c31a0
Serena account	https://goerli.etherscan.io/address/0xc6f1ce15d2473c0329ceba172f543ecdb323bbf3
Roger account	https://goerli.etherscan.io/address/0x57593698d6f7931debf144f5712cb090fe997d65
BC4C-ETH SC address	https://goerli.etherscan.io/address/0xf82646c6f578730f3d239e837de5eaacb08ee990
Task 1, T1	https://goerli.etherscan.io/tx/0x46057010418de7db617c745442386f3a475a420e8b1caa7fa4ede4c9992e097e
Task 1, T2	https://goerli.etherscan.io/tx/0xa4675fad4ec3efd53c7e5ed383e3883a5896586420613919961e90e657e28386
Task 1, T3	https://goerli.etherscan.io/tx/0xca43bed4a8e9b3f9f7f3d9d96b676725fa728c8fd7a3109b0aa891acebe39ca2
Game ID	4 digits id of the game created: 2032
Task 1, T4	https://goerli.etherscan.io/tx/0x9120c6052f5412c8dd51dfecb482ebc7575735f1f8bc60699ffd98219eaeb574
IEA1 Contract address	https://goerli.etherscan.io/address/0xdc4f52305b57e50326cf2d140e7fa0c835304ca2
Task 2, T1	https://goerli.etherscan.io/tx/0xa9b331d4bd72dad8b71d3a3612189bd8bdd9b45d35695ee10dcef292f4596421
Task 2, T2	https://goerli.etherscan.io/tx/0xdfb37e2b7648c38db9329e6000d4cabd01b27eabcbe9abf7062444113e8d599b
Task 2, T3	https://goerli.etherscan.io/tx/0xeba460b9a9a93668984c5ca5824cbe847e5d39cbcffd40fa392ddd925ed44469
Task 2, T4	https://goerli.etherscan.io/tx/0x882aad62c2001b68dab5956b0f00af0930fcec5984f73dcab1d9bf7effa0eded
Task 2, T5	https://goerli.etherscan.io/tx/0x750ea876245d5541713dcc3d28836a9bd18e444451e0c803ce51d3b72e0905a5
Task 2, T6	https://goerli.etherscan.io/tx/0x45f3fbf0c54838cb57ee83ff93e663b74338c4b8372df9d71874216bf7c0ac53
Task 2, T7	https://goerli.etherscan.io/tx/0xbbd2d3d4fc98fbe55de7af9f41c6615c0373c42fcfe3c046c82e1741c9b870cd
Task 2, T8	https://goerli.etherscan.io/tx/0xc217613e47f4c0bb284bb55a7bd65f3238d419456f36142c71f435c801266d8f

Note:

1. When I first deployed my smart contract for Task 2, the function **extendSupply()** first checked whether the payment amount was 0.009 ETH or not using an **if-statement** and only then executed the rest of the logic. I did it because in the assignment description for that function, it was said that the

token supply is extended **only if** 0.009 ETH are transferred to the token contract when calling this function and there was no mention about the error message when the condition is not satisfied. Then, since my code worked correctly when I tested it locally, I deployed the contract and executed all of the required transactions. However, a bit later, I realized that, when calling the extendSupply() function, if non-zero amount not equal to 0.009 ETH is transferred to the contract, the contract will still keep it, although the supply is not extended. I thought that it might not be desirable, and that is why I changed my implementation of the extendSupply() function by removing the if-statement and instead writing the **require()** condition with an error message, which guarantees that the function will execute the rest of the statements only when the condition is satisfied and otherwise it will throw an error message, and the inputted value won't be kept within the contract. After the changing the function, I deployed the new contract, and implemented all of the necessary transactions again.

2. I also changed the name of the function balanceOf() to balanceOfAccount() because of some reason (maybe it is not allowed for a variable and a function to have the same names) the function was giving an error at the compilation stage.

3. If you look at the Venus account, there is a transaction where I mined more ETHs from the faucet. I did it to ensure that if transaction or deployment fees get too high, I will have enough ETHs to do the necessary operations.