Indiviual Coursework

The individual course work consists of 3 parts

- Bitcoin Testnet Transaction (15 points)
- Smart contract for cash in DAML (20 points)
- Deploy a constant product AMM contract in Solidity (65 points)

Deliverables

Upload one PDF file that includes the following information:

- Complete Python code (BTC transactions) and smart contract code (DAML and Solidity);
- Screenshots of the transaction submitted;
- Answers and explanations requested below.

1. Bitcoin Testnet Transaction

Use the Python code developed in the lab practical session.

1. Create 4 Bitcoin Testnet addresses. Add below the addresses and the corresponding secrets, namely address1, address2, address3, address4. (2 points)
2. From a BTC testnet faucet send testnet bitcoin to one of the addresses. (1 point)
3. Create a 1-input 3-outputs transactions transferring 50%, 30%, 15% of the amount respectively into address2, address3, address4 respectively. (5 points)
4. Sign the transaction and submit it into the Bitcoin testnet via https://live.blockcypher.com/btc-testnet/pushtx/ (https://live.blockcypher.com/btc-testnet/pushtx/) (5 points)
5. How much fees are transferred to the miner and how are they calculated? (2 points)

2. Smart contract for cash in DAML

In this exercise you have to build a smart contract template in DAML. Please take screen shot of the code and transaction view of the script results.
To do this, you are asked to create a new contract template called `Cash`. In `Cash`, a bank (issuer) will print some `USD` and transfer to a New Holder. The New holder

updates the exchange rate, seeking to convert the currency from `USD` to `GBP`. Then, the Exchange checks the contract and do the swapping if the exchange rate is above the required threshold.

Contract template

1. Start by creating a new project and .daml file.
2. Create a contract template called `Cash`. In the contract you must specify datatypes for the following:

- amount:
- currency:
- issuer:
- holder:
- exchange:
- exchangeRate:

3. Then, define the roles of the parties. What type of party should the issuer be? And the exchange? (2 points)
4. Add a condition to ensure the amount of cash is larger than zero (2 points)
5. Add a function `Transfer` which transfer the cash to new holder, where the controller is the holder (2 points)
6. Add a function `UpdateExchangeRate` which sets a new Exchange rate, where the controller is the holder (2 points)
7. Add a function `Swap` which converts the currency (currency = newCurrency), updates the amount with the specified exchange rate (amount = amount/exchangeRate), and asserts the exchange rate is larger than 1.0 (exchangeRate > 1.0) (2 points)

Scenario testing

In the scenario testing part, using the following structure

```
cashTests: Script ()
cashTests = script do

--- Add parties

--- Create contract

--- Transfer the cash

--- Update the exchange rate

--- Swap the currency

return()
```

you must:

- Create three parties: “Party_1” (the issuer), “Party_2” (the holder), “Party_3” (the exchange).
- Let the issuer “Party 1” issue a new contract where the issuer “Party 1” wishes to transfer 100 `USD` to “Party 2”. At this stage set the issuer = the holder (2 points), and the exchangeRate = 0.0 (2 points)
- Let the holder (=issuer) transfer the cash to “Party 2”. (2 points)
- Let the new holder “Party 2” update the contract with exchangeRate = 1.2 (2 points)
- Let the exchange “Party 3” swap `USD` to `GBP` . (2 points)
- Try to let the holder do the swap. What will happen? Explain why this would happen. (2 points)

3. ERC20 and AMM Deployment

Your submission should include answers to the following questions.
You are recommended to read through all the questions first before you start the attempt.

1. Create two tokens with the ERC20 interface:
   name: `comp163_1` , `comp163_2`
   symbol: `comp1` , `comp2`
   decimals: 18
   totalSupply: 100 for each

Note: please make sure to read carefully the in-line comments and block comments. Also, read through the entire script to understand the structure of the code first before starting to attempt.

What are the contracts? (15 points)

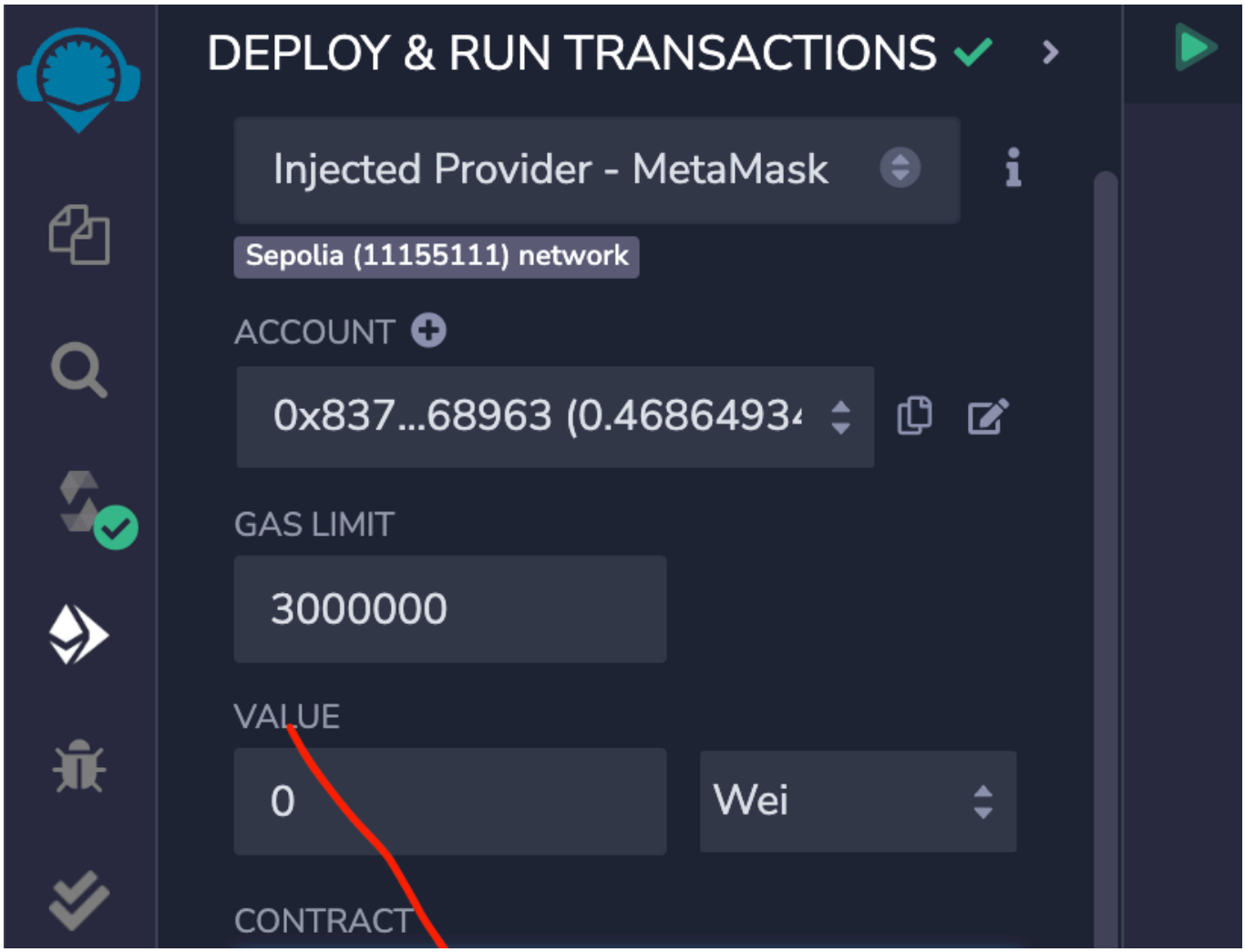
2. Test the ERC20 contracts by deploying it on the Remix VM. (Please take screen shot of your transaction overview)

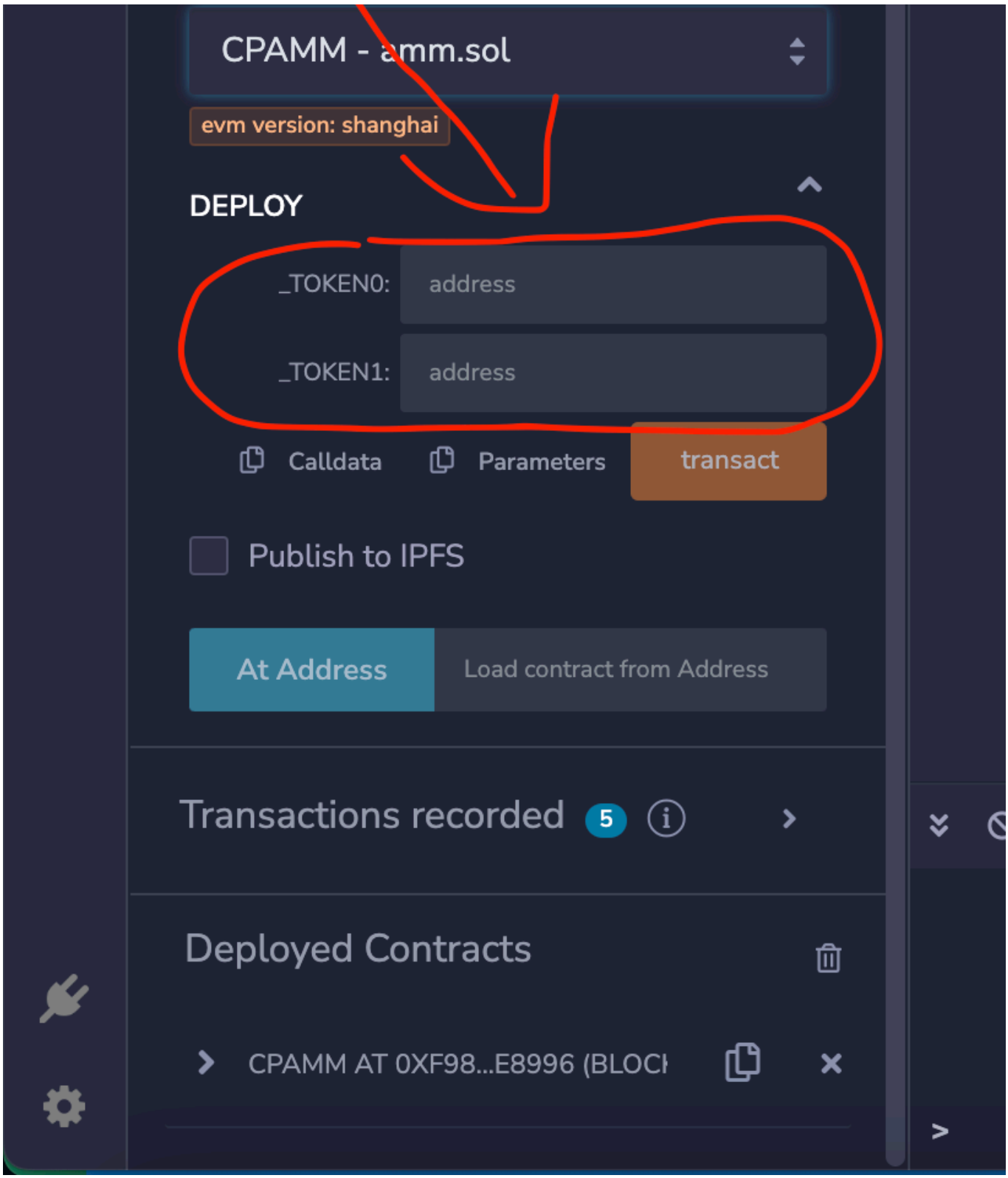
   What's the balance of `comp1` and `comp2` for your address? (5 points)

3. Create a simple constant AMM contract by replacing "____" with the actual codes.

   Note: please make sure to read carefully the in-line comments and block comments. Also, read through the entire script to understand the structure of the code first before starting to attempt.
   What is the complete contract? (20 points)

4. Test the AMM contract by deploying it on the Remix VM.
   What's the transaction hash of your creatation of the contract (5 points)





5. Approve 50 `comp1` and 50 `comp2` to the AMM contract, then add liquidity to the contract. Approve another 10 `comp1` to the AMM contract, then swap `comp1` for `comp2` . Remove the liquidity. (Please take screen shot of your transactions overview) (15 points)

6. Comparing the swap functions in question2 and question3, we can see that the AMM swap doesn’t need approval from a centralized financial agent. What other traditional financial agent could be replaced by the smart contract? (specify the agent and explain why) (5 points)