Homework 7

- Follow the instructions very carefully. Answers that do not conform to the instructions will not be given credit.
- Understand thoroughly all the code given to you in this lab. Search for documentation online if there is a primitive or API you have not encountered before.
- Read the Frequently Asked Questions at the end of this homework.
- Submit both your metacoin.js and MetaCoin.sol files. Do not zip them.

It's important to complete this part 1 very early so you don't run into last minute technical issues blocking you from completing the assignment on time. This part 1 should not take more than 3 hours to complete.

Part 1: Set up your environment

1. If you are using Windows: download and install cygwin, a Linux terminal emulator. Be sure to use the 64-bit installer if your machine is 64-bit. If you are using Mac or Linux, you don't need Cygwin -- you can just use the native terminal.
https://www.cygwin.com/.

2. Download and install VirtualBox, a free virtual machine.
https://www.virtualbox.org/wiki/Downloads

3. Download and install Vagrant. Read the getting started guide. Vagrant will make it very easy to get an Ethereum development environment set up.
https://www.vagrantup.com/downloads.html
https://www.vagrantup.com/intro/getting-started/index.html

4. Start up the virtual machine included in the homework 7 zip, log into it using ssh. Then, start the Ethereum node server, and then use truffle to test MetaCoin contract. Below are the step-by-step instructions. Two out of the four tests should pass.

Open a terminal

```
$ cd homework7
$ vagrant up
$ vagrant ssh
$ cd homework7/metacoin
$ ganache-cli
```

Open a new terminal

```
$ cd homework7
$ vagrant up
$ vagrant ssh
$ cd homework7/metacoin
$ truffle test
```

5. Confirm that the `~/homework7` directory on the virtual machine (VM) is the same as the `homework7` on the host. You can do this by opening windows explorer and creating a file under your unzipped `homework7` directory, and then confirming that same file is now created in the `~/homework7` directory on the VM. You can use your IDE to edit the `homework7` files from the host and your edits will be instantly reflected in the VM. This shared directory mechanism enables you to write code on the host, and test your code on the virtual machine without having to copy files between the host and the VM.

Part 2: Working with smart contracts

6. Read the Ethereum white paper and the documentation on solidity.
https://github.com/ethereum/wiki/wiki/White-Paper

https://solidity.readthedocs.io/en/develop/

7. Read the documentation on Truffle, an Ethereum testing framework that makes it easy to write contracts and test them on your own, private Ethereum network.
http://truffleframework.com/docs/

8. Read and understand the metacoin/MetaCoin.sol contract and unit test file metacoin.js. You can read about this contract in the Truffle documentation: http://truffleframework.com/docs/getting_started/contracts.

9. Find the empty function called "mint" in the MetaCoin contract. Give an implementation of the function that enables the owner of the contract to mint new coins and assign them to himself. Only the owner can mint coins. If anyone else calls the function, the function should not do anything.

10. Find the two failing test cases in metacoin.js and fill them in. One test should check that the owner can mint coins to himself. The other test should check that a non-owner cannot mint coins to himself.

Frequently Asked Questions

Question: The Truffle compiler keeps giving me unusual errors such as “Error: Invalid number of arguments to Solidity function” even though I am sure my code is correct.

Answer: Try deleting the build directory first, and then run “truffle test”. Truffle is a relatively new framework and has some quirks.

Question: My test case uses the truffle “call” method on a contract function that updates a contract state variable. However, when I call the state variable after the update to check its value, I find that the state variable did not get updated.

Answer: The truffle “call” method does not have an effect on the contract storage—it should only be used to read state variables. Invoke the contract function directly in the truffle test case in order for the function to have an effect on contract storage. For example, instead of `sendCoin.call(account_two, amount, {from: account_one})`, use `sendCoin(account_two, amount, {from: account_one})`.