Introduction to programming: HSG-Chain

An initiative by Thomas, Joël, Marc, Manfred, Armin and Raphael

In this project, we developed a block chain system called HSG Chain, the system is composed of two main classes:

Block Class:

This class represents the element of every block chain, every block is characterized by sequence identifier (seq_id), the block hash(block_hash), the hash of the previous block (prev_hash), a dataframe of transactions (transactions), the status of the block (Committed or Uncommitted) to the block chain and the merkle root.

To respect the OOP we defined those variables as private, so to access them we provided their getters and setters.

Other members functions:

display_header : This function uses the getters of the class to display the information of the block.

add_transaction(s,r,v) : updating the class transaction dataframe, with a new transaction, this function needs 3 parameters : a sender(s), a recipient (r) and the value of the transaction (v).

display_transactions: displays all the transactions saved in the block.

set_simple_merkle_root(): this setter is used when a block exceeds the allowed limit of transactions, calculate merkle hash by taking all transaction hashes, concatenate them into one string and hash that string producing a "merkle root", this is just a simple implementation of **merkle root**.

HSGChain Class:

This class will be managing the blocks, by creating a new block every 10 transactions and saving the old one to a list of blocks. This class contains the name, id, seq_id, prev_hash and current_block which is the block in use by the HSGChain class.

Members function:

display_chain: This method should loop through all committed and uncommitted blocks and display all transactions in them. This method uses **display_transactions** from block class.

add_transaction(s,r,v): this function checks if the current block contains 10 transactions (max number of transaction by block). If not the new transaction will be added to the current block. If the number exceeds 10, the current block will be committed and a new one will be created, the transaction will be added to the new block.

commit_block(block): This function is responsible for the following changes:

1- Changing block status from Uncommitted (default status) to Committed.

2- Creating the merkle root of the block using member function **set_simple_merkle_root()** from block class.
3- Creating the block hash by hashing a string combination of (chain id, timestamp, sequence id, previous block's hash and merkle root).
4- Add the block to list of committed blocks.
5- Create a new block.
6- Increment

display_block_headers: display information about blocks (committed and uncommitted). This function uses the **display_header()** member function from block class.

get_number_of_blocks: return the number of blocks (committed and uncommitted).

get_values: get all transactions value of the whole HSGChain system.

Transaction/User input: We currently hardcoded 25 random transactions. Of course, this is a random number and can be changed within the code (line 228). When the user starts the blockchain, he is asked if he wants to add manual transactions:

"Do you want to add a transaction (y/n)" y=yes; n=no

Then the user can insert the **sender**, **recipient** and the **amount of HSG-Coins**.

Those manual transactions are then added at after the random transactions (e.g.: when there are 25 random transactions, your first manual transaction starts at 26^{th} position)

Some properties of the Blockchain that can be easily changed are:

Difficulty: Line 11

Number of transactions per block: Line 48

Number of randomly generated transactions: Line 228

Random transactions are generated: Line 228 (setting the line to comment)

Distribution of the randomly generated transactions' Coin amounts: Line 208

For detailed explanations see the comments in the code.

"HSG-Coin: Get rich or die tryin'"

-Anonymous-