Name: Erfan Rasti

Student Number: 9823034

Report of 2nd Homework of Advanced Programming C++

GitHub Repository HTTPS:

ErfanRasti/AdvancedProgramming-HW2 (github.com)

In this homework, we are going to implement two classes Server and Client.

Server class is more complicated and more important.

We start with the member functions of the Server class:

- **get_client member function**

  In this member function, we are going to iterate between elements of a map block and find our desired id. If the desired id isn't found, we return "nullptr".

- **Add_client member function**

  In this member function, we add a new client to member value clients. If this new client is repetitious, we add a 4 digit random number to the end of the class using the default random engine and uniform real distribution. We find repetitious id using "get_client" to prevent duplicate code.

  We initialize the credit of each client with 5 coins.

- **get_wallet member function**

  At the first of this function, we find the desired client according to the given id using the "get_client" function. Then we return the mapped element to the specific client.

- **parse_trx static member function**

This function should be static because the only way we could call a member function without defining an object is by using a static member function.

Three inputs “sender”, “receiver” and “value” of this function should be defined with references. Otherwise, we cannot change the values of these variables.

In this function, we use function “get_line” from namespace std on library “sstream”. At first, we convert the input transaction string to a string stream. Then we create an array with 4 string blocks. The 4^{th} block is for detecting the error of extra dashes(more than 2). Using a for loop we split each element and copy that to the array “trxElements”. If the string is inputted correctly, the value of “i" in the last iterate of for loop is 3. We check this value and return an error if it’s not 3. At last, we assign the 3 first values of the array to the variables “sender”, “receiver” and “value” and return a true value for determining successful operation.

- **add_pending_trx member function**

This member function adds a transaction to the pending transaction list.

First, we define three main variables related to the desired transaction. Using the “parse_trx” member function we change the values of these variables. If the parse transaction operation succeeds we check if the sender and receiver are clients of our server. Then we check the wallet of the sender to make sure that he has enough credit. At last, we check the signature input and if it is valid we add the transaction to the pending list and return true. Otherwise, we return false.

- **mine member function**

  First, we build the “mempool” string by concatenating strings of vector “pending_trxs”. Then we start mining. The mining process will be continued until we find a miner. We iterate between clients and in each for loop we take a nonce value from each client using the “generate_nonce” member function. Then we check if the sha256 value of concatenating “mempool” and our nonce has the string “000” in the first 10 digits of itself. If it has, we found a miner. We give this client 6.25 credit and start pending transactions. At last, we clear the pending transaction list and we return the nonce value.

- **show_wallet function**

  In this function, we are going to access a private member value. We should make a pointer to the block of memory that the clients are. Out input object is server but we want to access clients. So we should cast this input using “reinterpret_cast”. We don’t use this casting format as usual. But in this special case, we have no choice.

Let’s explain Client class:

- **Client Constructor**

  In this function, we initialize the client object and generate public and private keys belonging to that client using function “generate_key”.

- **get_id member function**

  This is a simple member function that returns the id of the client.

- **get_publickey member function**

  This function returns “public_key” generated in the constructor.

- **get_wallet**

We return the credit of the client by accessing the server of the client.

- **sign member function**

This function returns the signature of the input transaction using the “crypto” library and “signMessage” function.

- **transfer_money member function**

We concatenate the string of receiver id and value of transaction with the current client(sender). Then we sign the transaction using the sign member function. At last, we try to add this transaction to the pending list. If this operation succeeds we return true. Otherwise, we return false.

- **generate_nonce member function**

We use the same engine that we used in the “add_client” member function in the “Server” class to generate a random number lower than 10^9. Then we return this value as a nonce.

Note: I used some “cout” functions in the code for figuring out the flow of data, managing memory, and optimizing the member functions.