به نام خدا

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

گزارش تمرین ۲

محمد برآبادی ۹۷۲۳۰۱۵

- فایل server.h :

```cpp
#ifndef SERVER_H
#define SERVER_H

#include <iostream>
#include <map>
#include <memory>
#include <random>
#include <vector>

class Client;

class Server
{
public:
    Server();
    std::shared_ptr<Client> add_client(std::string id);
    std::shared_ptr<Client> get_client(std::string id) const;
    double get_wallet(std::string id);
    static bool parse_trx(std::string trx, std::string &sender, std::string &receiver, double &value);
    bool add_pending_trx(std::string trx, std::string signature);
    size_t mine();
private:
    std::map<std::shared_ptr<Client>, double> clients;
};

inline std::vector<std::string> pending_trxs;


#endif //SERVER_H
```

در ابتدا کتابخانه server.h را تعریف میکنیم.

- فایل Client.h :

این کتابخانه هم برای client میباشد.

```cpp
#ifndef CLIENT_H
#define CLIENT_H

#include <iostream>
#include <random>
#include <memory>
#include <string>
#include "crypto.h"
#include <map>

class Server;

class Client
{
public:
    Client(std::string id, const Server& server);
    std::string get_id();
    std::string get_publickey() const;
    double get_wallet();
    std::string sign(std::string txt) const;
    bool transfer_money(std::string receiver, double value);
    size_t generate_nonce();
private:
    Server const* const server;
    const std::string id;
    std::string public_key;
    std::string private_key;
};
```

```
inline void show_wallets(const Server &server)
{
    std::map<std::shared_ptr<Client>, double> *c{(std::map<std::shared_ptr<Client>, double> *)&server};
    std::cout << std::string(20, '*') << std::endl;
    for (std::map<std::shared_ptr<Client>, double> :: iterator i = c->begin(); i != c->end(); i++)
    {
        std::cout << i->first->get_id() << " : " << i->second << std::endl;
    }
    std::cout << std::string(20, '*') << std::endl;
}

#endif //CLIENT_H
```

- فایل server.cpp:

در این تابع قصد داریم که client جدیدی اضافه کنیم و اگر آن اسم وجود داشته یک عدد چهار رقمی رندم جای آن قرار میگیرد.

```
#include "server.h"
#include "client.h"

Server :: Server():clients{}
{

}
////////////////////////////////////////////////////////////////////////////
std::shared_ptr<Client> Server :: add_client(std::string id)
{
    std::string idd = id;
    std::random_device rd;
    std::default_random_engine generator(rd());
    std::uniform_int_distribution<> distribution(1000,9999);
    for(auto i = clients.begin();i != clients.end();i++)
    {
        if(i->first->get_id() == id)
        {
            int number = distribution(generator);
            idd += std::to_string(number);
            break;
        }
    }
    Client a(idd, *this);
    std::shared_ptr<Client> b = std::make_shared<Client>(a);
    clients.insert({b,5});
    return b;
}
```

در تابع get_client اسم را میدهیم و clientموردنظر را برمیگرداند.

```
////////////////////////////////////////////////////////////////////////
std::shared_ptr<Client> Server :: get_client(std::string id) const
{
    std::shared_ptr<Client> j{nullptr};
    for(auto i = clients.begin();i != clients.end();i++)
        if(i->first->get_id() == id)
            j = i->first;
    return j;
}
////////////////////////////////////////////////////////////////////////
double Server :: get_wallet(std::string id)
{
    double j{0};
    for(auto i = clients.begin();i != clients.end();i++)
        if(i->first->get_id() == id)
            j = i->second;
    return j;
}
////////////////////////////////////////////////////////////////////////
bool Server :: parse_trx(std::string trx, std::string &sender, std::string &receiver, double &value)
{
    size_t find1 = trx.find('-');
    size_t find2 = trx.find('-', find1 + 1);
    if(find1 == std::string::npos or find2 == std::string::npos)
        throw std::runtime_error("We cannt find (-)");
    sender = trx.substr(0, find1);
    receiver = trx.substr(find1 + 1, find2 - find1 - 1);
    value = std::stod(trx.substr(find2 + 1));
    return 0;
}
```

در تابع get_wallet هم مقدار موجود در کیف پول مربوط به ID را برمیگرداند.

در تابع parse_trx هم وقتی trx وارد میشود به فرستنده و دریافت کننده و مقداری که بین این دو جابه‌جا میشود.

```
////////////////////////////////////////////////////////////////////////
bool Server :: add_pending_trx(std::string trx, std::string signature)
{
    std::string sender{}, receiver{};
    std::shared_ptr<Client>  temp1{}, temp2{};
    double value{}, value_sender{};
    parse_trx(trx, sender, receiver, value);
    temp1 = get_client(sender);
    temp2 = get_client(receiver);
    value_sender = get_wallet(sender);
    bool authentic = crypto::verifySignature(temp1->get_publickey(), trx, signature);
    if(authentic && temp1 != nullptr && temp2 != nullptr && value_sender >= value)
    {
        pending_trxs.push_back(trx);
        return true;
    }
    return 0;
}
```

```cpp
//////////////////////////////////////////////////////////////////////////////
size_t Server :: mine()
{
    size_t size{pending_trxs.size()};
    std::string mempool{}, sender{}, receiver{};
    std::shared_ptr<Client>  temp1{}, temp2{};
    size_t nonce{};
    double value{};
    for(auto i: pending_trxs)
        mempool += i;
    while(1)
    {
        int i{0};
        parse_trx(pending_trxs[i], sender, receiver, value);
        temp1 = get_client(sender);
        nonce = temp1->generate_nonce();
        std::string hash = crypto::sha256(mempool + std::to_string(nonce));
        if(hash.substr(0, 10).find("000") != std::string::npos)
        {
            std::cout<<sender<<std::endl;
            clients[temp1] += 6.25;
            break;
        }
        i++;
        if(i == size)
            i = 0;
    }
    for(auto i:pending_trxs)
    {
        parse_trx(i, sender, receiver, value);
        temp1 = get_client(sender);
        temp2 = get_client(receiver);
        clients[temp1] -= value;
        clients[temp2] += value;
    }
    pending_trxs.clear();
    return nonce;
}
```

- فایل client.cpp:

```cpp
#include "client.h"
#include "server.h"

Client :: Client(std::string id, const Server& server):id{id},server{&server}
{
    crypto::generate_key(public_key, private_key);
}
///////////////////////////////////////////////////////////////////////////////
std::string Client :: get_id()
{
    return id;
}
///////////////////////////////////////////////////////////////////////////////
std::string Client :: get_publickey() const
{
    return public_key;
}
///////////////////////////////////////////////////////////////////////////////
double Client :: get_wallet()
{
    Server s = *server;
    return s.get_wallet(id);
}
///////////////////////////////////////////////////////////////////////////////
std::string Client :: sign(std::string txt) const
{
    std::string signature = crypto::signMessage(private_key, txt);
    return signature;
}
```

```cpp
//////////////////////////////////////////////////////////////////////////////
bool Client :: transfer_money(std::string receiver, double value)
{
    std::string sign1{};
    bool i{};
    std::string trx{};
    trx += id;
    trx += '-';
    trx += receiver;
    trx += '-';
    trx += std::to_string(value);
    sign1 = sign(trx);
    Server s = *server;
    i = s.add_pending_trx(trx, sign1);
    if(i)
        return true;
    return 0;
}
//////////////////////////////////////////////////////////////////////////////
size_t Client :: generate_nonce()
{
    std::random_device rd;
    std::default_random_engine generator(rd());
    std::uniform_int_distribution<> distribution(0,8585);
    int number = distribution(generator);
    return static_cast<size_t>(number);
}
```

https://github.com/MBW0lf/AP_HW2

با تشکر