Hw(14.09.22)

Необходимо определить класс ***User*** (функции объявлены, названия говорят сами за себя, поэтому не буду здесь приводить лишние описания).

```
class User{
    private:
        std::string m_name;
    public:
        User();
        User(std::string name);
        void setName(std::string name);
        std::string getName();
        ~User();
};
```

Класс ***User*** нужен нам для корректной работы класса ***SocialNetwork***, который, (сюрприз?!) тоже надо будет полностью определить. Основной смысл объекта класса ***SocialNetwork*** заключается в имитации игрушечной социальной сети (функционал для упрощения сильно урезан, однако концептуально сущности те же).

```
class SocialNetwork{
    private:
        std::map< User, std::vector<std::string> > m_friends;

    public:
        SocialNetwork();
        SocialNetwork(const SocialNetwork & other);
        bool addUser(const User );//проверка, если юзер существует и добавление
        //1 - если добавился, 0 - если нет.
        void befriend(const User & us, std::string who);
        void unfriend(const User & us, std::string who);
        std::vector<std::string> getFriend(const User & us);
        void printSN() const;//распечатать всю социальную сеть в таком формате
        //имя юзера --- список его друзей через запятую.

        bool deleteUser(User &  us);//Задача сложного уровня!
        //удаление юзера. Имейте в виду, что нужно удалять еще и юзера у всех из друзей!
        ~SocialNetwork();//нужно ли освобождать память???
};
```

В качестве единственного поля данных выступает словарь (map) ***m_friends***. Где ключом является юзер (человек), а значением – вектор из имён его друзей (заметьте, вектор не из ***User*** обьектов, а из имён --- std::string, такое

упрощение сделано намеренно, несет в себе некоторые плюсы и минусы-подумайте, какие именно).

Конструктор копирования должен полностью копировать всю базу юзеров и их друзей.
Пример:

```
int main()
{
    SocialNetwork VK(FaceBook);

}
```

Функция **befriend(…)** должна добавить юзеру **us** в друзья человека по имено ***who*** ,стоит учитывать, что в таком случае, если в системе есть человек по имени ***who*** то у него в друзьях тоже должно появиться имя нашего ***us***.

Похожая ситуация с функцией ***unfriend()*** которая должна учитывать обоих юзеров, которые перестали дружить.

std::vector<std::string> getFriend(const User & us); должна выводить вектор с именами друзей юзера ***us.***

void printSN() const; должна распечатывать в консоль всю нашу социальную сеть в таком формате:

Имя юзера --- имена его друзей через запятую.

bool deleteUser(User & us); сложная функция. Должна удалять юзера, если он есть в списке ключей и из друзей всех юзеров, у которых он в списке друзей есть.

~SocialNetwork(); - деструктор.

Объявление данных классов есть на гитхабе в папке 30/hw/. Также ,имейте в виду, что список функций объявлен нестрого. Если у вас возникает необходимость определить вспомогательную функцию – делайте это со спокойной душой ☺.