学号：2022141410193　**姓名**：王枭越 **学院**：计算机学院

Assignment 4

保证我的工作文件夹为以下结构：

```
|
|--include--*.h
|--src--*cpp
|   |---users.txt
|
|--Makefile
|--README.md
```

代码

`login.h`

```
#include <string>
#include <vector>
#include <iostream>
#include <fstream>
#include <algorithm>
struct User{
    User(std::string name, std::string pass, std::string mail);
    ~User();
    std::string username;
    std::string email;
    std::string password;

    //----------------- other variables. you can add variables in this area
    };

struct Login {

    Login();

    ~Login();

    void readFile(std::string path); // reads the .txt file, then register the user
inside readFile using theregisterUser function


    bool checkUsername(std::string &new_username); // checks if the user is already
taken, if so, it returns true and the username shouldn't be created
    //and prints "username already taken"

    bool checkEmail(std::string &new_email); // only new emails may be pushedback to
"emails" vector

    void changeUsername(std::string username, std::string newUsername);
```

```
    void changePassword(std::string username, std::string newPassword);

    void changeMail(std::string username, std::string newMail);

    void registerUser(std::string username, std::string password, std::string
email); // function to creat a user

    void loginUser(std::string username, std::string password);

    void removeUser(std::string username);

    User* getUser (std::string username); // returns the usernames using the same
email, returns false if that email doesnt exist



    std::vector<User*> users;
    std::vector<User*> LoginedUsers;


};

/*
a@gmail.com,ali,1234
a@gmail.com,alireza,1234   DOESN'T CREATE A NEW EMAIL
b@gmail.com,alireza,1234   DOESNT CREATE A NEW EMAIL NOR NEW USERNAME AS "alireza"
IS ALREADY TAKEN
c@gmail.com,reza,4444
c@gmail.com,reza007,4444
c@gmail.com,reza11,4444

*/
```

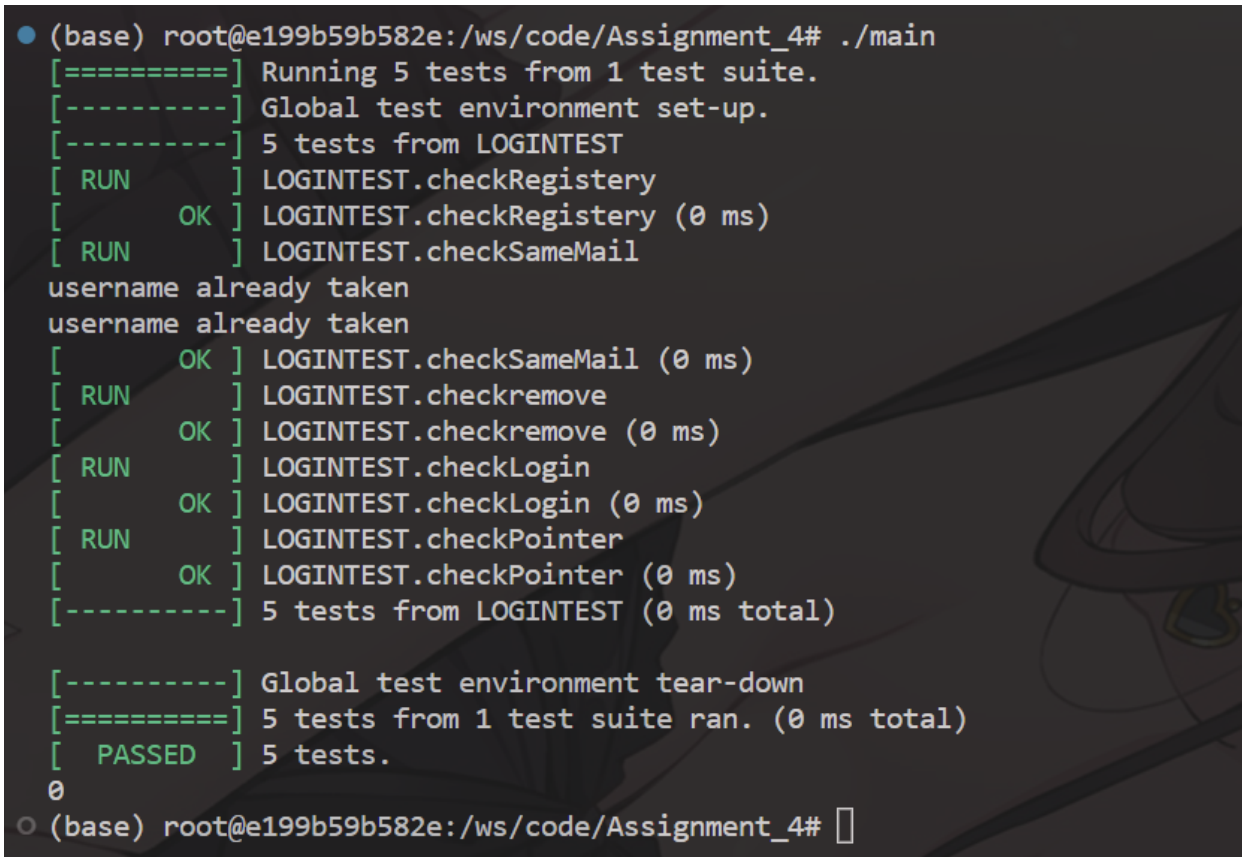
`login.cpp`

```
#include "../include/login.h"

User::User(std::string name, std::string pass, std::string mail){
    username=name;
    password=pass;
    email=mail;
}
User::~User(){

}

Login::Login(){

}
```

```
Login::~Login(){
    users.clear();
    LoginedUsers.clear();
}

void Login::registerUser(std::string username, std::string password, std::string
email){
    if(!checkUsername(username)&&!checkEmail(email)){
        User *p=new User(username,password,email);
        users.push_back(p);
    }
}

bool Login::checkUsername(std::string &new_username){
    for(int i=0;i<users.size();i++){
        if(new_username==users[i]->username){
            std::cout<<"username already taken"<<std::endl;
            return true;
        }
    }
    return false;
}

bool Login::checkEmail(std::string &new_email){
    for(int i=0;i<users.size();i++){
        if(new_email==users[i]->email){
            return true;
        }
    }
    return false;
}

void Login::loginUser(std::string username, std::string password){
    // for(int i=0;i<users.size();i++){
    //     if(users[i]->username==username&&users[i]->password==password){
    //         LoginedUsers.push_back(users[i]);
    //         break;
    //     }
    // }
    User *p=getUser(username);
    if(p!=nullptr){
        if(p->password==password){
            LoginedUsers.push_back(p);
        }
    }
}

User* Login::getUser(std::string username){
    for(int i=0;i<users.size();i++){
        if(users[i]->username==username){
            return users[i];
        }
```

```
    }
    return nullptr;
}

void Login::removeUser(std::string username){
    User *p=getUser(username);
    if(p!=nullptr){
        users.erase(find(users.begin(),users.end(),p));
        if(find(LoginedUsers.begin(),LoginedUsers.end(),p)!=LoginedUsers.end())
        LoginedUsers.erase(find(LoginedUsers.begin(),LoginedUsers.end(),p));
        delete p;
        p=nullptr;
    }
}

void Login::changeUsername(std::string username, std::string newUsername){
    User *p=getUser(username);
    p->username=newUsername;
}

void Login::changePassword(std::string username, std::string newPassword){
    User *p=getUser(username);
    p->password=newPassword;
}

void Login::changeMail(std::string username, std::string newMail){
    User *p=getUser(username);
    p->email=newMail;
}

void Login::readFile(std::string path){
    std::ifstream in(path);
    std::string tmp,username,password,email,islogin;
    std::getline(in,tmp);
    while(in>>username>>password>>email>>islogin){
        registerUser(username,password,email);
        if(islogin=="yes"){
            loginUser(username,password);
        }
    }
}
```

实现结果如下



下面简要说明思路：

1. 构造函数：直接赋值初始化。
2. 析构函数：只有`Login`需要自定义析构函数（其实struct自动释放），实现就是调用`vector`的析构函数释放。
3. `bool Login::checkUsername(std::string &new_username)`的实现：遍历查找即可。
4. `bool Login::checkEmail(std::string &new_email)`：同上思路。
5. `void Login::registerUser(std::string username, std::string password, std::string email)`的实现：在满足不重复的条件下用`new`语句去分配一个`User`同时初始化即可。
6. `void Login::loginUser(std::string username, std::string password)`的实现：分配一个`User`并观察密码是否正确，如是，则压进登录`vector`中。
7. `User* Login::getUser(std::string username)`的实现：遍历查找即可。
8. `void Login::removeUser(std::string username)`的实现：将`Users`中的删除，若登录还要将其从`Logineduser`移除。
9. `void Login::changeUsername(std::string username, std::string newUsername)`的实现：用`getUser`查找并更改即可。
10. `void Login::changePassword(std::string username, std::string newPassword)`的实现：同上。
11. `void Login::changeMail(std::string username, std::string newMail)`的实现：同上。
12. `void Login::readFile(std::string path)`的实现：用`iofstream`读入即可，同时注册，再判断是否登录再使其登录。

至此，所有问题解决。