Viết chương trình bắn tàu online:

Client	Server
Chức năng kết nối từ Client đến Server	
- ***connect [ip server] port [port server]*** : Client kết nối đến server với ip server và port server đang mở.	- Chấp nhận kết nối và tạo thread mới để gửi nhận dữ liệu với client.
- ***close*** hoặc ***end***: Client kết thúc phiên kết nối với server.	- Đóng kết nối và hủy thread với client (Ghi chú: Server vẫn hoạt động bình thường)
Chức năng đăng nhập và quản lý thông tin người dùng	
- ***login [username]***: login server với tài khoản của người dùng; sau dòng này hiện lệnh nhập password (chú ý: password không được hiện ra khi người dùng nhập). Khi đăng nhập, sẽ hiện ra yêu cầu có bảo mật tài khoản người dùng khi gửi dữ liệu cho Server không? Nếu có: thực hiện mã hóa username và password của người dùng, sau đó gửi chuỗi đã mã hóa cho server; Nếu không: gửi username và password dưới dạng chuỗi (không sử dụng thuật toán mã hóa). **ví dụ 1:** login legon >> password: ******* Do you want to encrypt message before sending? (Y/N): Y Login successfully or Wellcome legon login to server and Message was encrypted. **ví dụ 2:** login legon >> password: ******* Do you want to encrypt message before sending? (Y/N): N Login successfully or Wellcome legon login to server and Message wasn't encrypted.	Nhận username và password từ phía Client. Thực hiện kiểm tra thông tin sau: - username không tồn tại trong Database → thông báo lỗi và gửi message phản hồi cho client. - username tồn tại và password không đúng → thông báo lỗi và gửi message phản hồi cho client. - username tồn tại và password đúng → thông báo đăng nhập thành công.
- ***register [username]***: đăng ký tài khoản mới, sau dòng này hiện dòng bắt người dùng nhập password (chú ý: password không được hiện ra khi người dùng nhập). Khi đăng ký tài khoản mới, sẽ hiện ra yêu cầu có bảo mật tài khoản người dùng khi gửi dữ liệu cho Server không? Nếu có: thực hiện mã hóa username và password của	Nhận username và password từ phía Client. Thực hiện kiểm tra thông tin sau: - username đã tồn tại trong database → thông báo lỗi và gửi message phản hồi cho client.

người dùng, sau đó gửi chuỗi đã mã hóa cho server; Nếu không: gửi username và password dưới dạng chuỗi (không sử dụng thuật toán mã hóa). **ví dụ 1:** register legon >> password: ******* Do you want to encrypt message before sending? (Y/N): Y Register successfully and Message was encrypted. **ví dụ 2:** register legon >> password: ******* Do you want to encrypt message before sending? (Y/N): N Register successfully and Message wasn't encrypted.	- username không tồn tại trong database → thông báo đăng ký thành công, lưu tài khoản mới vào database và gửi yêu cầu client đăng nhập.
- ***change_password***: yêu cầu đổi password mới của user hiện hành, sau lệnh này bắt người dùng nhập password cũ, nếu password cũ nhập đúng thì hiện dòng tiếp theo yêu cầu nhập password mới (chú ý: password không được hiện ra khi người dùng nhập). Khi thay đổi password, sẽ hiện ra yêu cầu có bảo mật tài khoản người dùng khi gửi dữ liệu cho Server không? Nếu có: thực hiện mã hóa password của người dùng, sau đó gửi chuỗi đã mã hóa cho server; Nếu không: gửi password dưới dạng chuỗi (không sử dụng thuật toán mã hóa). **ví dụ 1:** change_password >> password: ******* Do you want to encrypt message before sending? (Y/N): Y >> new password: ******* Change password successfully and Message was encrypted. **ví dụ 2:** change_password	Nhận password từ phía Client. Thực hiện kiểm tra thông tin sau: - password cũ không đúng → thông báo lỗi và gửi message phản hồi cho client. - password cũ đúng → nhận password mới từ phía Client → cập nhật password mới trong database.

>> password: ******* Do you want to encrypt message before sending? (Y/N): N >> new password: ******* Change password successfully and Message wasn't encrypted. **ví dụ 3:** change_password legon >> password: ******* >> new password: ******* Do you want to encrypt message before sending? (Y/N): N [Server] Your password is wrong, Plz give it again. >> password: ******* >> new password: ******* Do you want to encrypt message before sending? (Y/N): Y Change password successfully and Message was encrypted.	
- ***check_user [-option] [username]***: kiểm tra thông tin của một người dùng khác (Ghi chú: chỉ sử dụng một option), bao gồm các option sau: o ***–find***: kiểm tra tài khoản người dùng có tồn tại? o ***–online***: kiểm tra tài khoản người dùng có online? o ***–show_date***: hiện ngày sinh của tài khoản người dùng (dd/mm/yyyy) o ***–show_fullname***: hiện tên của tài khoản người dùng o ***–show_note***: hiện ghi chú của tài khoản người dùng o ***–show_all***: hiện tất cả thông tin cá nhân của tài khoản người dùng o ***–show_point***: hiện số trận thắng của người dùng **Ví dụ:** check_user –online Hans >>User is online check_user –show_date Hans	Nhận lệnh từ phía Client và thực hiện kiểm tra: - username không có tồn tại trong database → thông báo lỗi và gửi message phản hồi cho client - username có tồn tại trong database → kiểm tra option và thực hiện truy vấn trong database.

<table>
<tr><td>>> Birthday of Hans is 07/08/2000</td><td></td></tr>
<tr><td>- setup_info [-option]: thiết lập hoặc thay đổi thông tin cá nhân của người dùng hiện hành (đang đăng nhập), gồm có các option sau (Ghi chú: chỉ sử dụng một option):
o –fullname [Chuỗi tên]: tên của người dùng
o –date [birthday]: ngày sinh của người dùng (dd/mm/yyyy)
o –note [chuỗi ghi chú]: ghi chú của người dùng
Ví dụ: setup_info –fullname “Nguyen Van A”
>>Name of Hans is “Nguyen Van A”
setup_info –date 07/08/2000
>>Birthday of Hans is 07/08/2000</td><td>Nhận lệnh từ phía Client và thực hiện các lệnh cập nhật trong database.</td></tr>
<tr><td colspan="2">Chức năng tham gia trò chơi</td></tr>
</table>

Client 1	Client 2	Server
start_game : lệnh bắt đầu trò chơi ***>> create_room [id room] with [user_name]:*** mở một phòng chơi mới với người dùng chỉ định. **Ví dụ:** start_game [Server] List users is online: Hans Paul Ken….. *//Nhận dang sách user đang online từ phía Server* >> create_room 52 with Hans	Nhận thông báo tham gia chơi game với Client1 từ phía server. Hiện thông báo có chấp nhận trò chơi không? **Ví dụ:** >> Do you join game? (Y/N)	Nhận lệnh yêu cầu chơi game từ phía Client1 Gửi danh sách user đang online cho Client1 Gửi thông báo cho Client2 và chờ phản hồi từ Client 2 - Nếu Client2 chấp nhận: Mở một phòng chơi giữa 2 client với nhau - Nếu Client 2 từ chối: Phản hồi lại cho Client1
>> upload_ships [file upload] : upload vị trí tàu lên server từ file. (dùng file text hoặc execl)	*//tương tự, thực hiện upload vị trí tàu lên server*	Nhận vị trí tàu từ phía 2 Client và thông báo lượt đi đầu tiên cho Client1
>> attack_ship A B : tấn công tại vị trí dòng A và cột B Ví dụ: Your turn: >> attack_ship 3 4 Nhận thông báo vị trí tàu của bản thân và tàu của bạn bị loại từ phía server	*//tương tự, chờ và nhận lượt đi của mình* Nhận thông báo vị trí tàu của bản thân và tàu của bạn bị loại từ phía server	Nhận tấn công từ phía Client1 tại vị trí AB, kiểm tra: - Nếu không có tàu: thông báo cho Client1 và báo lượt đi cho Client2 - Nếu có tàu: loại bỏ tàu tại vị trí AB. Thông báo vị trí loại cho Client1 và Client2 Cập nhật liên tục vị trí tàu bị loại cho 2 Client.

Nhận thông báo kết thúc trò chơi. Nhận kết quả thắng/thua. Hiện bản thông báo tiếp tục game? (Y/N)	Nhận thông báo kết thúc trò chơi. Nhận kết quả thắng/thua. Hiện bản thông báo tiếp tục game? (Y/N)	Thông báo trò chơi kết thúc cho 2 Client khi số tàu của 1 Client bị bắn hạ hết hoàn toàn Nhận thông báo tiếp tục game từ 2 Client: - Cả 2 đồng ý: tiếp tục trò chơi (từ vị trí upload_ships và attack_ship) - 1 trong 2 không đồng ý: kết thúc trò chơi và đóng room của 2 Client

Kích thước bản đồ và tàu:

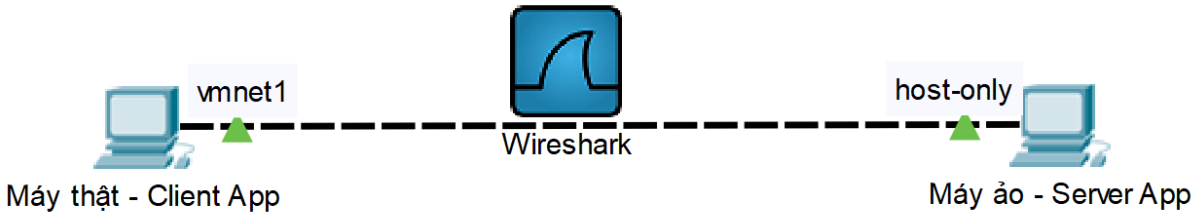
- Kích thước bản đồ: 20x20
    - Số lượng và kích thước tàu:

○ 3 tàu 1x1	○ 3 tàu 1x2	○ 3 tàu 1x4	○ 2 tàu 2x5	○ 1 tàu 2x7

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1						x													x	
2																			x	
3																				
4				x	x				x	x	x	x	x							
5				x	x				x	x	x	x	x							
6				x	x										x	x	x	x		
7				x	x															
8				x	x															
9				x	x			x												
10				x	x									x	x					
11														x	x					
12														x	x				x	
13								x						x	x					
14								x						x	x					
15								x												
16			x					x												
17			x																	
18			x												x	x				
19			x																	
20									x	x										

Yêu cầu:

- Xây dựng ứng dụng console cho client với các dòng lệnh đã mô tả trên.
- Xây dựng ứng dụng console cho server, thực hiện nhận lệnh từ client và xử lý rồi gửi thông điệp về cho client.
- Sử dụng lập trình đa tiến trình (multi thread) cho mỗi client kết nối đến server.
- Sau khi xây dựng xong chương trình, dùng chương trình wireshark bắt gói tin trao đổi giữa server và client theo mô hình mạng sau:



- Báo cáo gồm có các phần:
  - Phần 1: cách xây dựng protocol, gồm có:
    - Kịch bản chương trình
    - Cấu trúc chương trình, bao gồm (không copy toàn bộ code):
      - Giao thức thực hiện: TCP/UDP.
      - Các hàm chính: mô tả rõ ý nghĩa tên hàm, giá trị truyền vào.
      - Kiểu cấu trúc: mô tả ý nghĩa kiểu cấu trúc dữ liệu đã xây dựng và thuộc tính của cấu trúc đó.
  - Phần 2: bắt và phân tích gói tin.
    - Chụp hình lại gói tin đã mã hóa và chưa mã hóa.
    - So sánh và cho biết khác biệt giữa hai dữ liệu này.
    - Cách làm:

Bảng so sánh gói tin	
Dữ liệu mã hóa	***Dữ liệu chưa mã hóa***
Chức năng Login	
<Ảnh gói tin>	<Ảnh gói tin>
Chức năng …..	
<Ảnh gói tin>	<Ảnh gói tin>

Hướng dẫn

Phần Mã hóa

- Có thể sử dụng thuật toán chuyển đổi dữ liệu: từ text sang bit, hoặc từ text sang hex … và ngược lại.

Đoạn code demo:

```
string string_to_hex(const string& in) {
    stringstream ss;

    ss << hex << setfill('0');
    for (size_t i = 0; in.length() > i; ++i) {
        ss << setw(2) << static_cast<unsigned int>(static_cast<unsigned char>(in[i]));
    }

    return ss.str();
}

string hex_to_string(const string& in) {
    string output;

    if ((in.length() % 2) != 0) {
        throw runtime_error("String is not valid length ...");
    }

    size_t cnt = in.length() / 2;

    for (size_t i = 0; cnt > i; ++i) {
        uint32_t s = 0;
        stringstream ss;
        ss << hex << in.substr(i * 2, 2);
        ss >> s;

        output.push_back(static_cast<unsigned char>(s));
    }

    return output;
}
```

- Có thể sử dụng các thuật toán mã hóa và giải mã (Encrypt và Decrypt) sẵn, và nhớ ghi chú lại nguồn sử dụng.