For your first project, you will implement a system in which exactly two clients can chat with each other over the Internet using TCP/IP.

Clients cannot contact each other directly; rather, they are paired up by a server process that may be running on any Internet host. If one client wants to chat with another client, it sends a request to the server containing (at least) the client’s chat nickname. If this is the first request that the server has seen, it will wait for another request. When a second request arrives, the server will notify both clients of the other’s nickname. Once a client receives a response, it can chat with the other client. Messages between clients are relayed by the server; clients cannot contact each other directly.

The server should run until one client closes its connection. When this happens, all 3 processes should exit gracefully, closing their sockets.

Your client program should accept one parameter on the command line: the host on which the server is running. The port on which the server is listening should be set as a constant in a shared header file.

In addition to your code, I will expect a detailed README document (just a plain text file, please) that explains your design and communication protocol, the rationale for your design decisions, and how your system works. The purpose of this is twofold. First, it will help me to understand your code and your design. Second, it is meant to make you think carefully about your design, hopefully before you start coding.

The grade will be based both on test cases and a read of your code. I will be looking for elegance and efficiency, in addition to correctness.

**Phase 1:** In the first phase, assume client messages are strictly alternating. In other words, client 1 sends, then client 2, then client 1 again, etc. The server will decide who goes first and communicate this to the clients before chatting begins. Your output should precede each outgoing and incoming message with the nickname of the author. Successful completion of this phase will earn you 85%.

**Phase 2:** In phase 2, augment your system so that clients can send and receive asynchronously. In other words, a client can send multiple times in a row and can receive messages while typing an outgoing message. In a single terminal window, this may look a little messy, but try your best to make it readable. (You may also want to look into the ncurses library, which gives you more control of terminal window. There is a book in the department library.) Your implementation must not contain any polling; while either process is waiting for activity, it should be blocked. Successful completion of this phase will earn you 95%. (Hint: D & C, 5.1–5.3, 5.5.)

**Phase 3:** To earn all possible credit, augment your system further with other capabilities. I will leave these to your imagination...