Today you will implement the classic 'mail' command of UNIX that was (once upon a time) used by most of us to send emails. 'mail' is a program that negotiates a connection with a server and sends an email using SMTP (Simple Mail Transfer Protocol). SMTP is a simple text protocol. The following messages are sent between a client and an SMTP server on port 25 in the following order:

1. Client sends the "EHLO" command with an argument equal to the hostname of the machine initiating the connection on port 25.
2. Server sends an ack or a reject
3. Client sends the MAIL command with the return address for the message, which has the following form *mail to: <email.address>*, and is ended by \n.
4. Server sends an ack or a reject
5. Client sends the RCPT command with the recipient address for the message, which has the following form *rcpt to: <email.address>* and is ended by \n.
6. Server sends an ack or a reject
7. Client sends the "DATA" command to signal the beginning of the message text.
8. Server sends an ack or a reject (If it can receive the message or not)
9. The message contain one or more lines of text. To terminate the message, the last line must be a period with a single linefeed. The message is terminated with: .\n
10. Server sends an ack or a reject (If it accepts or rejects the methods)

To make this extra clear, a transcript of such a conversation while using "telnet" as the client is shown in Figure 1. Note how the conversation starts with a 'EHLO' message whose argument (in quotes) is the name of the host from which the email is being sent. The server responds with a message of several lines reporting its capabilities. The next message of the client is 'mail from:' with the email address of the sender. The server responds with one line, saying (in essence) 'Ok'. The next client message is the 'rcpt to:' message with the email address of the recipient. This prompts (again) a one line answer from the server. The client wraps up by sending 'DATA' on a line all by itself, followed by multiple lines of text containing the body of the message. A final line with a period (and nothing else) followed by a line-feed ends the email, and the server responds indicating that the message was queued. The client then terminates the exchange with a 'quit' message. The UNIX mail command (that you are implementing here) automates this whole process. You run it by invoking `mail` with the address of the recipient on the command line. Your mail program uses your user identifier and your domain to form the address of the sender. `mail` should prompt the user for the body of the email and deliver it to the server. Clearly, this involves the creation of a socket to contact the server and sending the right messages over the socket.

In addition to the protocol, you should implement the socket send and receive functions.

Note: Keep in mind that when reading from and writing to sockets, the basic write() and read() functions may not write the entirety of the payload. Your functions should ensure that all of the payload is written and read from the socket.

```
[jforce@jforce ~]$ telnet smtp.uconn.edu 25
Trying 137.99.25.233...
Connected to smtp.uconn.edu.
Escape character is '^]'.
220 mta3.uits.uconn.edu ESMTP Postfix (Debian/GNU)
EHLO jforce
250-mta3.uits.uconn.edu
250-PIPELINING
250-SIZE 31457280
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
mail from: jordan.force@uconn.edu
250 2.1.0 Ok
rcpt to: tyler.daddio@uconn.edu
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
subject: Hello tyler
Hey tyler,

This is an email test for the lab! Should be fun!

--Jordan

.
250 2.0.0 Ok: queued as C274A883
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

Figure 1: Transcript of SMTP session over telnet. My commands are in black; server response is blue