1. The implementation strategy mostly consisted of 2 large parts: the socket api calls, and the string parsing. I began with writing the client socket api calls as they were the simplest to familiarize myself a bit more with the socket api. Then I wrote the string parsing for the sserver portion as it was the most complex. Then I wrote the sserver socket api calls as they were a bit more complex, and then reused the string parsing for the client-side header reads. The concurrent connections were done using method a in the assignment description, using fork. I had a while loop with accept to grab the next connection in queue (automatically handled by accept()), and I had a queue size of 10, as was recommended in Beej (5-10 connections).
2. I tested the string parsing by writing it inside the client code, and then passing the file-to-read.txt buffer into the string parsing so that way I could test things such as case-insensitivity and switching around header values. I tested the connections mostly just by running them and having a lot of print statements that would tell me when things are progressing properly.
3. Currently I believe the bugs in my code have to do with adding some extra newline characters in some spots that aren't supposed to be there. For some reason, when sending a message back and forth, the client and server code both seem to add new line characters before and after the message. I haven't had the time nor energy to debug this since I've been ill, unfortunately.
4. I mostly collaborated with Joshua as we discussed the implementation of the code. I also used the Beej sockets guide to help me through the socket api.