Important Note: For step 5, the phrase “username not found” or “username found” is printed at the end of the terminal. Thus, the phrase will be printed on the 3rd line from the bottom in the terminal. To make it easy to find this print statement, I intentionally printed one new line (“\n”) before and after the phrase. This will make it easier to spot the response for step 5.

1. Team details: Palak Mehta (pm862), Harini Vujini (hv95)
2. Collaboration: While we did not collaborate with anyone else for this project, we did use TAs as a source to clarify any questions that came up along the way for us. For example, we joined office hours where possible to run our thought process on the code by the TA so we knew exactly what was expected of us. For example, since part 5 was slightly confusing, we made sure to join office hours in order to clarify it. We asked if we needed to address the idea of usernames exceeding 8 characters or if that is already taken care of. We also sent emails to TAs with any questions we had. Other than this, we used a reliable website called https://www.geeksforgeeks.org/difference-between-stop-and-wait-gobackn-and-selective-repeat/ on the internet to make sure we had a clear understanding of the difference between selective repeat and pipeline reliability. Since we already knew the concepts from earlier in the semester, we just needed to make sure that we were approaching the code correctly. It also took a little bit to understand the structure of the messages that we were getting. For example the message class has 4 important attributes and all we really needed to use was the sequence, message, and ack fields. After this investigation, we knew exactly what my output should be. Overall, since we had discussed this at length via homeworks and lectures, we wanted to make sure that we weren't simply coding to follow what the instructions told us to output, but that we were actually understanding the concepts we were trying to implement.
3. Our code works as described.
4. We did encounter some difficulties along the way. For example, when working on part 4 of the project, we noticed that despite having pretty solid logic within the code, we were seeing small differences come up between my input and output files when we used the “diff” command. Everytime we ran the code, only the first message was not sending properly. We tried various troubleshooting strategies such as specific print statements to try and understand where the code could be going wrong in sending/receiving packets. Then, finally we decided to re-read the instructions to see if we missed anything. At this point, we realized the “ooo_enabled” flag was missing in my command and we took a moment to understand the importance of it in ensuring the packets are sent in an appropriate order. While that was a less technical challenge, it still helped us understand the importance of the flag. When it came to part 5, we found that we were having a difficult time figuring out the correct way of approaching the

solution since there was no “TODO” comment in the receiver file. In order to figure out the appropriate place to begin editing the receiver file, we looked for where the packets were being read and outputted in the file to begin with. After we found that position, it became much easier for us to embed my logic of part 5 within it. From here, we ensured that the first packet contained the logic for the username.

5. Overall through this project, we learned about the distinction between timers in selective repeat as opposed to stop-and-wait. The idea is that in stop-and-wait an ack must be sent for each packet in order for the next one to arrive, while in the selective repeat, there is the idea of a cumulative ack. The cumulative ack stores packets that are out of order or packets after the packet that is lost. Thus, the pipeline reliability is able to transmit all the data a lot faster. We know in theory that this is true from lecture, however seeing it in practice is cool. The packets were transmitted a lot faster in part 4 than in part 5. This was interesting to keep in mind because even though our implementation remained consistent with the select method for both, there still is a difference between the two. In addition, we learned about the implementation of “windows” via the transmit_entire_window_from() method. The idea of sliding both the left edge and right edge based on the size of the window and ensuring that they didn’t exceed the final sequence number was very interesting. It also helped me in my visualization of the concept of windows. This distinction in part 4 demonstrated how logically cumulative acks work in practice. It is easier to keep track of the window size by using these two pointers. Overall, this project was very helpful in understanding the major difference between selective repeat and stop-and-wait.