Report for Project #4: HTTP server with authentication

1. Team Members:

   Yuchen Wei (yw780) and Tiange Zhang (tz196)

2. Collaboration:

   We use GitHub to collaborate between the two of us, updating and communicating the parts we have completed and the changes and improvements we have made to the code. The primary reference materials we used in the process were from the lectures we received, the previous Python programming courses we participated in, and our experience in learning Python independently. We completed sever.py, stopandwait.py and report together offline.

3. Is there any portion of your code that does not work as required in the description above? Please explain.

   After comparing the results of our tests with the samples included in the project and the description, we believe that our code fulfills all the requirements of the description and can be reflected by the tests.

4. Did you encounter any difficulties? If so, explain.

   When we started trying to finish sever.py after successfully completing stopandwait.py, we initially thought we could just update both left_edge and right_edge and use the transmit entire window from(x) method to complete the transfer successfully. However, after testing and running the program, we found that according to the log, although the transfer was complete, the program kept trying to transfer the last packet's ack without stopping automatically. We then solved this problem by adding a flag of size INIT_SEQNO + content_len and automatically breaking when final_ack equals it.

5. Describe two technical observations or facts you learned while working on this project. Please answer in specific and precise terms.

Real-world cases of packet/ack losses and loss handling are much more complicated than what we learnt in the lecture. Although cases of such losses in this project are implemented with a certain pattern of loss or a fixed probability, but it gives us a basic idea of how to deal with losses. We start with a stable but slow transmission method (stop and wait), and then switch to a faster transmission one. And from the given files, we could clearly see each when loss happens and specializes and how a duplicate package is retransmitted.

Another observation we learnt is the implementation of the transmission window. It is quite a challenge to fully understand the edges and the window size. We noticed how the given file determine and handle the right edge when the content length is smaller than the default window size, which could've resulted in an error or inefficiency.