We ran programs using “python [Program Name]”, not python3

1. Naman Bajaj – nb726
   Roshan Aiyar – ra854

2. In terms of people, we only collaborated and discussed ideas with each other. We consulted Python’s socket documentation for information on how to use select.select in addition to the Piazza, and TA slides, which were the main sources we used for reference on how to set up the programs, in addition to Project 1.

3. We used select.select which works by taking in 4 arguments, however we only need to focus on the first and the fourth. The first argument is the list of sockets that we want to read in information from (in this case, ts1 and ts2), and the $4^{th}$ argument is a timeout argument, which waits that number of seconds before moving on. Select works by querying the list of sockets simultaneously, so it meant we could wait for ts1 and ts2 to return something, rather than asking ts1, waiting 5 seconds, then asking ts2 and waiting another 5 seconds. It was especially useful that select.select stops blocking once any of the sockets return something, so it meant we could move on once either ts sent a value. Select.select returned a tuple of 3 lists, however we only focused on the first value in the tuple, which was the results, which we then checked if they were empty or not, and based on that, sent a value back to the user.

4. We did a fair amount of testing and everything seems to work as asked, including sending information between different machines.

5. We had difficulties with case insensitivity, since we were asked to return the value that ts had, rather than what the client originally sent in. We were able to fix this by changing the way we stored values in our dictionaries, that is we stored [Domain all lower, List[domain from ts text file, IP]] rather than [Domain, IP]. We also had some difficulty using select.select, as there wasn’t much documentation and we had to go off what was being said on Piazza, slides, and testing.

6. We learned how we can use load balancing to ensure that client requests are answered effectively and efficiently. It gave us a broader view on how requests in the real world can be sent to multiple locations before getting a response. In terms of specifics, we learned of the many ways that Python allows us to run tasks simultaneously, as we didn’t immediately decide to use select, we experimented with run blocking and multithreading before settling for select.