For this assignment, I used Redis 7.0.8 and Python 3.10. I used the default storage engine. My computer is a 2020 Macbook air with the Apple M1 chip and 16 GB RAM. Specific libraries I used were functools and collections.

API Method	**API Calls Per Second**
read_load_tweets	11248.6
load_timeline	19263.5

The main factor that influenced how many timelines I was able to get per second was that I used a sorted set instead of a list. I first architected the home timelines to all be lists that I would constantly push and pop from. However, my performance improved a bit when I switched to a sorted set.

There are a few factors that contributed to the performance of posting tweets. First, I used lru_cache to store each user's followers in memory as I was repeatedly getting users' followers to put each tweet into those followers' home timelines. This improved the efficiency by about 2000 tweets per second. Additionally, I constructed the timelines in reverse order to avoid adding and removing tweets posted very early on. I had planned on putting the update_timelines() function in a multiprocessor that would update timelines continuously in reverse order while new tweets were being read in. However, I was not able to achieve this because the multiprocessing library has a quirk where it does not work with objects because of AuthentificationStrings which cannot be pickled. Later on I might see if I could overwrite the Process object to change authentication keys to temporarily be stored as bytes, which would allow for the multiprocessing described above. Doing so would greatly improve the performance of posting tweets. However, as my code currently stands, posting tweets is not quite optimal.

On a more general note, I tried to use list comprehensions anywhere I could to improve efficiency.

Finally, my code assumes the redis-py folder or a reference path to it is in the same directory as the .py files, so you might have to change the location of the redis-py folder or add a reference to the redis-py folder via the terminal.