Jesse Berliner-Sachs

CIS 350 – HW 1 Analysis

1. I decided the easiest design would be a mainMenu function which would loop, constantly reading inputs until the program was quit. It would delegate the responsibility of each menu selection to a different function which did the processing (potentially using other helper functions) and output. Besides the main class I also created classes Tweet and State which did the heavy lifting for parsing the inputs from the input files. The each had fields for their respective attribute which corresponded for the most part to data points from the input files. These classes could also easily be created, stored and sorted by the respective functions. Both classes were comparable to help implement sorting in specific ways
2. My class design is good because
   1. Changing the format of the inputs is as easy as editing the regex strings within each class
   2. Adding data to the tweets (how many likes etc.) can be done by adding fields to the class
   3. Because the States were comparable they could be easily sorted.
3. My style is good because:
   1. Having classes for the States and Tweets make it understandable how data is parsed
   2. Having a separate function for each menu option makes it easy to see how the code relates to the interface
   3. My error codes are very descriptive so if something goes wrong when another programmer is modifying my code it is easy to see what the issue is
4. My program is efficient because
   1. The state file is read and preprocessed once instead of every time the states need to be read. This saves time when multiple menu options are used and doesn’t require much space since there are only 50 states.
   2. The Tweets are **not** preprocessed since there are ~150K of them which would require ~150K objects stored in memory which is much too large.
   3. I also made recursive elements for the program such as returning to the main menu using while loops instead of recursive function calls to limit stack space.
5. I tested my code by running it with different arguments and run-time inputs. I attempted (non-methodically) to test every function and line of code and on edge-cases. I paid special attention to invalid inputs and error handling to ensure my program didn’t crash.