Challenge Problem #2: In this challenge problem, you will implement a dictionary attack to reverse simple passwords hashed with the SHA512 algorithm. This will be similar to what was done for MD5 hashes on the first day of class. You will answer some additional questions to approximate run-time and study some complexity surrounding these attacks.

Directions:

1. Download the files `dictionary.txt` and `hash.txt` from Canvas, these can be found under 'Files/Challenge Problems' or on the Challenge Problem 2 assignment page.
2. Write a program in python, sage, or Google Sheets (veeeeery slow) which "reverse hashes" the four hashes in `hash.txt`. (That is, you get to hash each entry and compare the hashes until you find a match). Additionally, your program should track how long it takes to reverse each hash. (You can use a profiler, or you can just do something simple like capturing system time at the start and stop of program execution.)
3. The first two are in the dictionary exactly. The third is made up of two of the dictionary words separated by a space. The last entry appears in the dictionary, but the hashed version has both upper and lower case letters in an unknown combination.
4. **Warning:** Depending on how you write/implement your attack, the third and forth attacks could take quite some time to run (my machine can hash all case combinations of the provided dictionary in about 45 minutes, for example–but I'm on an old machine and I did nothing to optimize my algorithm, so ideally it shouldn't take you any longer than that).

Once you have reversed the hashes and timed your code, answer the following questions (you will turn in your responses along with your code for this assignment).

Assignment Questions:

1. What was the plaintext for each of the four hashes?
2. How long did it take your algorithm to reverse each hash?
3. For the third hash, how would the run time be altered if you knew one of the words, but not which position it was in? What is the maximum possible run time in that situation?
4. Based on your algorithm and what you found out, what is the minimum number of words (separated by spaces and appearing in the dictionary) you would need to use to guarantee a year to hash all combinations?
5. What are three things you could do to speed up the average run time of your algorithm? You do not need to implement any of these, but what are three things you think you could try to substantially impact the general run time in each case? I'm looking for design/implementation improvements or mathematical solutions, not "Run it on a faster computer".

Grading Criteria For this assignment you will be graded primarily on correctly reversing each hash (the first two can be easily done online, but alas, the later ones are unlikely to be in a standard reverse hash lookup) and having thoughtful answers to questions #3 and #4 showing a good faith effort in your responses. Writing quality (grammar and structure) should also be up to the standards outlined for mathematical writing.