Read Me

In the assignment, the main data structure is a hashtable of size (26^5) struct node pointers, which stays constant throughout the runtime of the program. For each unique word, it is hashed into the hashtable which uses linked lists to deal with collision. The space complexity of this is O(n) because the only thing that increases is the amount of key/value pairs in the hashtable which is exactly the number of unique words in the dictionary.

Because both the data and dictionary are read in letter by letter, those parts of the program will have a run time of O(km). The insertion of the dictionary words into the hashtable in the worst case will be O(m^2), simply because the hashtable might not be able to handle collisions well and therefore it would be the run time equivalent to adding to a linked list while keeping a sorted order. It is m^2 because each word attempts to be inserted into hashtable and if it is already in the hashtable then it is ignored, otherwise it is added. This is the worst case, but on average the hashing function distributes the keys wide enough that inserting each word takes closer to O(1) and in total for m words O(m). For checking the data words against the dictionary words, it takes worst case O(kn) for each data word, meaning total O(kmn). But because the data words are also hashed, the length of the linked list they need to check will not be n and instead be closer to 1 meaning the total runtime is O(km). Finally, the printing result has a runtime of O(n), because for every key/value pair in the hashtable there needs to be an output.

The biggest challenge of this assignment for me was figuring out a data structure that would allow me to not only to implement simpler algorithms while also having short runtime. It was also a slight challenge to deal with the logic for repeating and similar dictionary words.