Analysis

Part 1:

Based on our outputs and the multiple runs that we ran, we think that the HashMap data structure is more efficient than the TreeMap data structure when loading and searching. It is probably due to how each structure keeps elements. TreeMaps always keep elements in a sorted, increasing order while HashMaps have no order. It would take less time to load data into the HashMap since it does not have to be put in any specific order while TreeMaps do. For example, when we ran the loading time for HashMap, we got 697,700 nano-seconds while for TreeMap we got 2,201,000 nano-seconds. There was a significant difference between the two data structures and the HashMap was clearly faster. Also, since we are searching in random places in the data structure, it might be faster to look through an unsorted list since you could have a higher chance of finding something faster. For one of our runs, the search time for HashMap was 7,000 nan-seconds while for TreeMap it was 19,200 nano-seconds.

Part 2:

Similar to part 1, the loading time in the hash set was faster compared to the tree set. This means that hash sets are more efficient for inserting data compared to tree sets. Since hash sets are backed by HashMaps and tree sets are backed by TreeMaps, they have similar properties where hash sets are not in any order while tree sets maintain objects in sorted order. This would make the loading time for hash sets faster since there is no required order. Different from our part 1 results for searching, the tree

sets were faster compared to the hash sets. This may be a result of how big the file that was needed for part 2. Since there were more words to go through, it might have been better to search through something sorted rather than something unsorted. Therefore, tree sets were more efficient for searching data in this case.

Part 3:

According to our test run, ArrayList is more efficient at accessing random items than Linked List. However, Linked List is supposed to be faster at inserting items than an ArrayList based because the Arraylist has to push all of the items at the index and larger to the right when a new item is inserted however the result shows that the ArrayList(2949600 ns) is faster at inserting than a LinkedList(5169500 ns) when one item is being added to 9999 seperate lists. While Linked List only has to point to the new item. In addition, it takes about the same time to iterate through all the items in an ArrayList and in a LinkedList using the provided Iterator class. The time it takes to load items to each team (out of 9999 teams) and shuffle in an ArrayList and Linked List is about the same with the ArrayList at (3154140900 nano second) and LinkedList (3712754900 nano second). In our opinion, if you care about random accessing performance, you should use an ArrayList since it keeps indexes of all the items and therefore faster to access them. On the other hand, if you don't care about random access but inserting elements into the List, a LinkedList yields better performance.