1. The computational complexity of insertEnd() would be O(1) constant if you left a pointer at the tail. It could also O(n), n being the size of the Linked List, if you created an iterator and moved it to the end of the list each time the method was called. The computational complexity of insert() is O(n), n being the size of the Linked List in the worst case. This is because you have to iterate to a specific location prior to the insert. The iteration takes O(n), then insert takes O(1), for a total of O(n). delete() takes O(n) time because the pointer must be created and go from head to the index of deletion, which takes O(n), then delete at O(1), for a total of O(n). Edit() is O(n) because it needs to traverse to a specific location. Search() is also O(n) because it needs to traverse the entire list and make a comparison at each node. Print() would take O(n) because it has to traverse each node in the linked list and print each node's data.
2. Using a linked list as a line editor was just a challenge to test our abilities to use linked lists effectively. In the real world, no one would ever use a linked list for this because most of the functions require access of a specific index, which can't be done in constant time with a linked list. You could easily use a HashMap for this, since they can access indices in constant time. A HashMap would be a good data structure to use since you could order the list with keys, and access the individual data points by referencing those keys. Arrays can also access in constant time, but arrays would be bad for insert and delete if you would have to shift all the following elements to the left or right, which is not constant time. I would go with a HashMap. Advantages of LinkedLists are that you don't have to shift all the elements like in an array, you just deal with pointers, which is slightly easier somewhat. Printing and search are easy with Linked Lists because in both

methods you are required to touch every element in the data structure anyway, so that's fine. Search would be better if binary search was doable at O(log n).

3. This assignment taught me about garbage collection and memory handling in C++ that doesn't exist in Java, which is cool and interesting. This assignment also taught me about the differences between Java and C++ syntax when dealing with pointers. It also taught me that there are much better ways to solve certain problems if you can find and properly implement the right data structure. If I could start over I would constantly be moving a tail pointer to keep insertEnd at constant time. I'd also have tightened up any extra traversal or unnecessary comparisons to speed the runtime up.