**Computational Complexity:**

| Method | Computational complexity | Reason |
|---|---|---|
| addNode | O(n) | Need to traverse linked list until index n is reached to insert new node. |
| deleteNode | O(n) | Need to traverse linked list until index n is reached to delete specified node. |
| insertEnd | O(n) | Need to traverse entire list until last node is reached, so computational complexity depends on the size of the list. Calls addNode to insert a new node in position 1, which would be O(1). |
| edit | O(n) | Need to traverse linked list until index n is reached to edit the value of the specified node. |
| insert | O(n) | insert calls addNode and adds no additional complexity, so it is O(n). |
| deleteIndex | O(n) | deleteIndex calls deleteNode and adds no additional complexity, so it is O(n). |
| print | O(n) | Prints all n nodes, so computational complexity depends on size of list. |
| search | O(n) | Searches entire list for all instances of value to be searched, so computational complexity depends on size of list. |

**Advantages and disadvantages of using linked lists for implementing a line editor:**

Linked lists allow for O(1) insertion and deletion *if* an iterator is pointing at the desired position, but O(n) otherwise due to linear access -- you need to visit each node before moving on to the next. For the same reason, search, print, and edit are also O(n). The worst-case time complexities are a disadvantage, but linked lists are dynamic structures and work well in situations where the total number of nodes/values are unknown. Additionally, you can decrease the average case complexity by using iterators or creating a doubly linked list.

**Lessons learned from this assignment and what I would do differently:**

During this assignment, I learned how to implement and modify my own linked list, and I re-learned how to handle user input. I tried to create some basic methods that could be used within other methods, but I largely let go of that idea while I was debugging. I realized that my computational complexities were higher than necessary. If I were to start over, I would have used an iterator or established a tail, which would've reduced the complexity of many of my functions; for example, insertEnd could've been O(1).