TextEditor Commentary

***Complexity and Analysis***

* **INSERT END FUNCTION­**
  + This method goes thru the linked list and finds where the current node's next pointer is a nullptr (indicating its the last line of the doc) and makes a new line and sets the node's next pointer as the pointer to the new line --- O(n) time complexity when accessing the last line
* **INSERT FUNCTION**
  + This method finds goes thru the document a specified number of times until lineNm parameter variable is reached and makes a pointer to a new line object, and sets the previous nodes (lines) pointer to the new line pointer object --- O(n) time complexity (when going to the last line)
* **SEARCH FUNCTION**
  + This method goes thru the document and finds a substring specified by the text parameter, if found it prints the line number and line, otherwise prints "not found" --- O(n) time complexity; could go thru whole list
* **PRINT FUNCTION**
  + This method goes thru the document and prints the data at each "line" (node) ---O(n) time complexity; goes thru whole list
* **DELETE LINE FUNCTION**
  + This method goes thru the linked list until the specified line number (via for loop) and sets the prev next pointer to the line that the current line's next pointer points to and then deletes the current node --- O(n) time complexity when deleting the last line
* **MODIFY FUNCTION**
  + This method goes thru the linked list until the specified line number (via for loop) and sets the current lines data to the string specified by the text parameter --- O(n) time complexity; could go thru whole list
* **LINE CONSTRUCTOR**
  + This constructor checks for 80 character limit of a line, initializes the lines data to the string text parameter, and allows for initialization of the next pointer---O(1) time complexity because its only setting object variables to given parameters
* **DOCUMENT CONSTRUCTOR**
  + This constructor sets the head of the Document (a linked list) to the pointer specified as the head parameter--- O(1) time complexity because its only setting the object variable to the passed parameter

***Analysis of using a LinkedList for a Line Editor***

* **Pros**
  + Accurate modeling of relation of documents / the lines that it is comprised of
  + Easy to delete a “line” because the structure be mutable in length and the “lines” can still model the threaded functionality without much difficulty after a line is deleted
* **Cons**
  + Difficulty accessing (searching, editing, reading, etc.) lines (other than the head) because you have to go through each preceding line
* **General Thoughts**
  + Implementing a Linked List for a line editor, I think, is the most functional because it allows for relative ease of access of lines (which other linear data structure could you possibly use that wouldn’t have to go through each preceding line to access one in the middle?) and allows for easy threading during deletion of said lines.

***What I Learned***

Considering I knew linked list implementation from previous classes, this was more of a refresher on how to do so. I did learn how to comment more efficiently because, in re-reading of my past linked list implementation, I found it difficult to understand my code so implemented a new code style that’s more paragraph style. I learned quite a bit about error handling: checking to ensure that you’re avoiding null pointers. I also learned how to read in tokens/whole lines using the std namespace – the getline function and then parsing using string function substring. If I were to redo this project again I would do error checking not to the scope but just think about edge cases not necessarily given.