Data Structures (CS 1520) Homework #4 Due: March 3 (Saturday) at 11:59 PM

Objective: To gain experience with code reuse by reusing your previously implementing `CursorBasedList` data structures from homework #3 by calling its API/methods without modification.

To start the homework: Copy your **completed** hw3 directory to a new directory hw4.

Once you have your `CursorBasedList` class finished for homework #3, you are to write a simple text-editor program that utilizes your `CursorBasedList` class. When your text-editor program starts, it should ask for a text-file name (`.txt`) to edit. If the file name exists, it should load the file into an initially empty `CursorBasedList` object by reading each line from the file and use the `insertAfter` method to append the line to the list. **Each node in the list will hold a single line of the text file.** If the text-file name specified at startup does not exist, an empty `CursorBasedList` object is created to model editting a new file.

Regardless of whether you loaded a file or just created an empty list, a menu-driven loop very similar to the `cursorBasedListTester.py` program should allow you to edit the file's content by modifying the list. (**STRONG HINT:** You might want to start with the `cursorBasedListTester.py` program as a rough starting point for your text-editor program.) You should NOT need to modify your `CursorBasedList` class only create a `CursorBasedList` object and use its methods. Make sure that your editor does not violate any preconditions of the `CursorBasedList` methods, so your editor is robust, i.e., does not crash when editting. When done editing, the lines of data contained in the nodes of the `CursorBasedList` are written back to the text file.

Your text-editor program should present a menu of options that allows the user to:

- navigate and display the first line, i.e., the first line should be the current line
- navigate and display the last line, i.e., the last line should be the current line
- navigate and display the next line, i.e., the next line should become the current line. If there is no next line, tell the user and don't change the current line
- navigate and display the previous line
- insert a new line before the current line
- insert a new line after the current line
- delete the current line and have the line following become the current line. If there is no following line, the current line should be the last line.
- replace the current line with a new line
- save the current list back to a text file

Warning: When you load a text file into your list nodes, you can leave the '\n' characters on the end of each line of text. However, remember to add a '\n' character to end of inserted lines or replacement lines.

Implement AND fully test your text-editor program. Part of your grade will be determined by how **robust** your text-editor runs (i.e., does not crash) and how **user-friend**/intuitive your program is to use. You are required to submit a brief (i.e., about a page in length) User's manual on how to use your text-editor.

For extra credit, your program may provide do one or more of the following additional text-editor functionality:

- Find word and Find next occurance
- Replace a specified word/string on the current line by another word/string
- Copy and Paste a line, etc.

Be sure to include these additional features in your User's manual.

Submit all necessary files (cursor_based_list.py, node.py, node2way.py etc.) as a single zipped file (called hw4.zip) electronically at : https://www.cs.uni.edu/~schafer/submit/which_course.cgi