Problem Based Learning - Assignment 1

(Text Editor)

Objective

- Improve Problem-solving skill with Python and bridging the gap between learning the skill and putting it into practice.

Overview

This Assignment has 4 Parts:-

- **PART A**: Write Cursor based List Class.
- **PART B**: Write simple text-editor program which will be using your Cursor based List Class developed in Part A.
- **PART C**: Testing your developed program for its user, writing user manual for it and making it user friendly.
- **Bonus Part**

To start the homework:
- Download and extract the file PBL1.zip from **https://drive.google.com/file/d/11qyZ--DED6LUadVCjHT9H-ximXtSgZT_/view?usp=sharing** .

- The PBL1.zip file contains:
  - the Node class (in the node.py module) and the Node2Way class (in the node2way.py module)
  - the **skeleton** CursorBasedList class (in the cursor_based_list.py module) which you will complete with your coding skills
  - the cursorBasedListTester.py file that you can use to **interactively test** your CursorBasedList class.

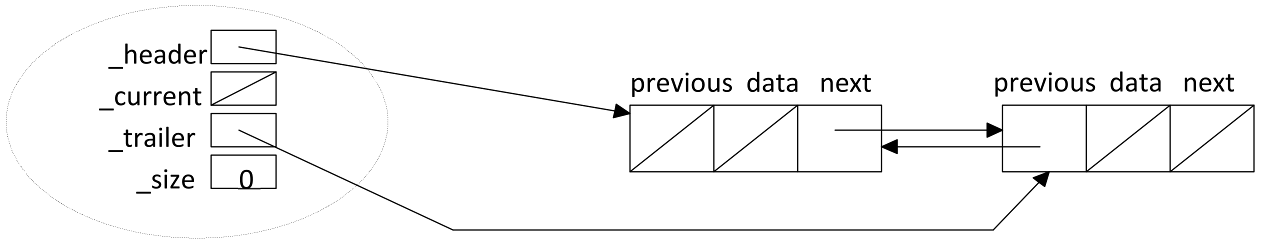
Details

In this assignment, your task is to create a text editor program using a cursor-based list and using a doubly-linked list with header and trailer nodes

- ***PART A: Write Cursor based List Class.***
  - In cursor-base list, a cursor (indicating the current item) can be moved around the list with the cursor being used to identify the region in the list to be manipulated. We will insert and removing items relative to the current item. A current item which is always defined as long as the list is not empty.

Cursor-based operations	Description of operation
`L.getCurrent()`	Precondition: the list is not empty. Returns the current item without removing it or changing the current position.
`L.hasNext()`	Precondition: the list is not empty. Returns True if the current item has a next item; otherwise return False.
`L.next()`	Precondition: hasNext returns True. Postcondition: The current item has moved right one item
`L.hasPrevious()`	Precondition: the list is not empty. Returns True if the current item has a previous item; otherwise return False.
`L.previous()`	Precondition: hasPrevious returns True. Postcondition: The current item has moved left one item
`L.first()`	Precondition: the list is not empty. Makes the first item the current item.
`L.last()`	Precondition: the list is not empty. Makes the last item the current item.
`L.insertAfter(item)`	Inserts item after the current item, or as the only item if the list is empty. The new item is the current item.
`L.insertBefore(item)`	Inserts item before the current item, or as the only item if the list is empty. The new item is the current item.
`L.replace(newValue)`	Precondition: the list is not empty. Replaces the current item by the newValue.
`L.remove()`	Precondition: the list is not empty. Removes and returns the current item. Making the next item the current item if one exists; otherwise the tail item in the list is the current item unless the list in now empty.

- The cursor_based_list.py file contains a skeleton CursorBasedList class. You MUST uses a doubly-linked list implementation with a header node and a trailer node. An empty list looks like:



- All "real" list items will be inserted between the header and trailer nodes to reduce the number of "special cases" (e.g., inserting first item in an empty list, deleting the last item from the list).
- Use the provided cursorBasedListTester.py program to test your list.

- **<u>PART B</u>: Write simple text-editor program which will be using your Cursor based List Class.**
    - You have to write a simple text-editor program that utilizes your CursorBasedList class. (You might want to start with the cursorBasedListTester.py program as a rough starting point.) Your text-editor program should present a menu of options that allows the user to:
        - enter a filename of a text (.txt) file to edit. Your program should then insert each line from the text file into a cursor-based list.
        - create a new text (.txt) file to edit. Your program should create an empty cursor-based list.
        - 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
        - *Provide additional text-editor functionality*: Find word, Replace word, Copy a line, Paste a line, etc. Be sure to include these additional features in your User's manual.

- ***<u>PART C</u>: Testing your developed program for its user, writing user manual for it and making it user friendly.***
    - 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 User's manual on how to use your text-editor.
- ***<u>Bonus Part</u>. For bonus your program may also do the following:***
    - Use a GUI package instead of a text-menu interface.

        - **Note:** you should still use the cursor-based-list ADT to store the lines of text as the editor runs.with a *graphical user interface* (GUI) and provide standard text editor functions.
        - For creating GUI, you can use any Python GUI framework and one of them is Tkinter package. Details at https://docs.python.org/3/library/tk.html)

- ***Tkinter.*** Tkinter is a Python Package for creating GUI applications. Python has a lot of GUI frameworks, but this is the only framework that's built into the Python standard library. It has several strengths; it's cross-platform, so the same code works on Windows, macOS, and Linux. It is lightweight and relatively painless to use compared to other frameworks. There are 3 steps in using Tkinter.
  a. Importing the Tkinter module.
  b. Creating a window in which the program executes. It is also known as the "root window".
  c. Finally, using a function to execute the code which is known as "mainloop()".

  Example:

```
# import all things from tkinter

from tkinter import *

# create root window

root = Tk()

# widgets,buttons,etc here

root.mainloop()
```

Submission

Submit all necessary files as per below mentioned details in a single zipped file at LMS a

1. Cursor_based_list.py (Output of Part A)
2. Text_editor_Main.py (Output of Part B)
3. UserManual.docx (Output of Part C)
4. Myerrorlogfile.txt
5. Node.py
6. Node2way.py
7. Plagiarism Report.pdf (20% Plagiarism allowed)

Submission Date

The due time for this assignment is 4 Jan 2021 (Monday) by 08:00 am via LMS. No extension on pretext of LMS issue. Complete your assignment and plagiarism check earlier than dead line to avoid LMS issue. Any delay will cost you marks.

Name your files with your name e.g Cursor_based_list.py should name as **Cursor_based_list _imranjavaid.py**. Zip all of your files for this assignment (Minimum 7 files), name the zip file yourname_section_Assignment1.zip e.g **imranjavaid_te56A_assignment1**.zip and submit the zip file at LMS. ***The bug fixing log file must follow the template given with assignment.***

Note:

- ***Each file has its own marks distrubition so submit all files.***
- ***All those who will be correctly following the submission procedure will get 1 BONUS MARK***

Grading policy **Total Mks : 100**

Plagiarism more than 20% OR fake Plagiarism report will result in ZERO (0). Rest of the assignments will be graded as per following policy:-

You will need to track of compiling errors which occurred during your development which **you fixed by yourselves** in a log. Each log entry should be different and minimum 5 log entries are necessary. ***Also write compiler/ IDE name in your log file***.

1	Correct Program with no compiler error	**(10 marks)**
2	Log for fixed compiler bugs	**(5 marks)**

- You will need to track compiling errors that you fixed by yourselves in a log for fixed compiler bugs that needs to have at least 5 entries for 2 points/each entry.

3	Complete Plagiarism Report (pdf)	**(10 marks)**
4	Cursor_based_list.py	**(20 marks)**
5	Text_editor_Main.py	**(40 marks)**
6	UserManual.docx	**(15 marks)**

Hints

1. Start the project as soon as possible.
2. You will need to create as many test cases as you can manage to thoroughly test your program. The final test cases to be used in the grading will be numerous.