Fall 2023

CS2001 – Data Structures

Assignment 02

Deadline: October 30th, 2023

Submission Information

You are required to submit your assignment on **Google Classroom**. The submission should consist of a single .zip file containing all of your code. It should be named in the following format:

DataStructures_Asg2_[Section]_[RollNo]_[Name].zip

e.g., **DataStructures_Asg2_BDS-3A_22I1234_Ali.zip**

Correct and timely submission of the assignment is the responsibility of every student. No relaxation will be given to anyone. For timely completion of the assignment, start as early as possible.

Your code must be generic and handle all errors and exceptions. If there is a syntax error in the code, zero marks will be awarded in that part of the assignment.

Plagiarism: Plagiarism is not allowed. If your assignment is found plagiarized, you will be awarded zero marks in the whole assignment category.

Learning Objective: The aim of this assignment is to have hands-on experience with linked list operations while building a practical application.

Implement a Console Based Notepad

Objective:

Create a console-based notepad application that allows users to **write** and **edit** text, incorporating functionality for **text manipulation**, **saving**, **loading**, and implementing **a spell-checker** using a provided dictionary.

Requirements:

1. **Console-Based Notepad Functionality:**
    a. Create a notepad that enables text input in a console environment.
    b. Allow users to add, delete, and edit text content.
        i. Text Deletion: Use the **"Backspace"** button to delete text.
        ii. Text Loading: Use **"Ctrl + L"** to load text from a file.

iii. Text Saving: Use **"Ctrl + S"** to save the output in the text file. You have to update the same file that you used to read the data.

c. The program will automatically save the data in file **"save.txt",** and load the data from the same file. There is no need to ask the user for the file name.
d. To quit the notepad, use the **"Esc"** button.
e. On program exit, release all memory occupied by the linked list data structure.
f. You should divide your screen into 2 distinct segments. **One for writing text in a notepad (75% of the screen) and one for displaying the correct alternatives for a word (25% of the screen).** This is the same idea that is used in Visual Studio. Most of the screen is reserved for writing code but some part of it is reserved for displaying error list.

2. Implement cursor control and tracking:

a. You have to implement your own cursor, which means the console will now have 2 cursors: the default cursor and the one that you would implement.
b. Your cursor can be moved using **arrow keys** to navigate through the text, and the default cursor will always stay at the end of the output, as expected.
c. In order to implement this functionality, you need to clear and redraw your output with the required changes on every cursor movement, to update the cursor position.
d. Normally the console requires the user to press the 'Enter' key to complete the input. In order to reflect the changes of cursor movement and typing, your program should **complete the input without pressing 'Enter' key.** One way to do this is to complete the input as soon as a key is pressed. That way you can store the letter typed by the user or any change in cursor movement as soon as a key is pressed, and update your output accordingly.

3. Data Structure:

a. Implement the notepad using a **two-dimensional doubly linked list**.
b. Its implementation is just like a doubly linked list with an additional property that it can grow in two dimensions.
c. Since text can be written on multiple lines, each row of the 2D linked list represents one line. Each line is terminated when a newline is inserted in the ADT.
d. Each node in the linked list has **four links (before, after, below, above)** and can store a character.
e. In addition, each node can store a character. Two words must be differentiated with a space character.

4. Dictionary:

You have to implement the functionality of a dictionary as well. Load the dictionary words from text file (provided to you) to your program. And save the words in a data structure.

For this assignment, you'll be using **Vectors** to store the words in the dictionary. Your program should read the text file on startup and store the words in a Vector. Your program should check any grammatical mistakes using the following methods.

5. **Spell Checker (Dictionary implementation)**

   You will write a simple spell-checker. For words that are not in the dictionary file, your program should suggest possible correct spellings by printing to the standard output console. You should perform the following modifications of a Misspelled word to handle commonly made mistakes:

   a. **Letter Substitution:** go over all the characters in the misspelled word, and try to replace a character by any other character. In this case, if there are k characters in a word, the number of modifications to try is 25*k. For example, in a misspelled word 'lat', substituting 'c' instead of 'l' will produce a word 'cat', which is in the dictionary.

   b. **Letter Omission:** try to omit (in turn, one by one) a single character in the misspelled word and see if that gets you a word in the dictionary. In this case, there are k modifications to try where k is the number of characters in the word. For example, if the misspelled word is 'catt', omitting the last character 't' will produce a word 'cat', which is in the dictionary.

   c. **Letter Insertion:** try to insert a letter in the misspelled word. If a word is k characters long, there are 26*(k+1) modifications to try, since there are 26 characters to insert and k+1 places (including in the beginning and at the end of the word) to insert a character. For example, for 'plce', inserting 'a' after 'p' produces a correctly spelled word 'place'.

   d. **Letter Reversal:** Try swapping every 2 adjacent characters. For a word of length k, there are k - 1 pairs to swap. For example, in 'paernt', swapping 'e' and 'r' produces a correctly spelled word 'parent'.

On the allocated space (25% of whole console), print out each misspelled word together with all candidate correct spellings that you found in the dictionary.

For example, if your dictionary file contains only the following six words:

cats, like, on, of, to, play,

and the file to spell check contains 4 words: **"Catts lik o play"**

the output should be:

catts => cats

lik => like

o => on, to, of

Notice that the list of possible correct spelling must contain only unique words. In the example above, for the misspelled word 'catts', removing the first 't' or the second 't' leads to the same word 'cats', but 'cats' should appear in the output only once.

If no modification of the word produces a correctly spelled word, your program should report **"You are a hopeless speller!"**

Note:

It is explicitly prohibited to declare any kind of array for any purpose in the implementation.

Marking Rubric

The assignment will be marked as follows:

Task		**Marks**
Data Structure		**10**
	Correct structure definition	10
Notepad Functionality		**70**
	Cursor Control and Tracking	25
	Text Addition	10
	Text Editing	10
	Text Deletion	10
	Text Saving	5
	Text Loading	5
	Quit Notepad	5
Interface (screen division)		**20**
Dictionary (Vector Data Structure)		**10**
Spell Checker		**40**
	Letter Substitution	10
	Letter Omission	10
	Letter Insertion	10
	Letter Reversal	10
Correct Output		**10**
Total		**160**