National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	DATA STRUCTURES - LAB	Course Code:	CL2001
Program:		Semester:	Summer 2024
Duration:	2 Hours	Total Marks:	100
Paper Date:		Weight:	
Section:		Pages:	2
Exam:	Final		

Student : Name:_____ Roll No._____ Section:_____

Instruction/Notes:

- We will check your code for plagiarism. If plagiarism is found, it will result in an F grade in the lab.
- In case of any ambiguity make a suitable assumption.
- No cell phones are allowed. Sharing of USBs or any other items is not allowed.
- You are not allowed to have any helping code with you.
- The path for submission is:

Question 1: Custom Text Editor Implementation for Smart Keyboard Application **[Marks - 40]**

Note: Use Array based data structure

You've been assigned to develop a next-generation smart keyboard application that revolutionizes the way users interact with text on their devices. This application integrates advanced AI algorithms to predict and adapt to user input, enhancing productivity and streamlining text editing tasks. As part of this project, you're responsible for implementing a crucial feature: a simplified text editor that supports three fundamental operations - INSERT, BACKSPACE, and UNDO. The goal is to create a function in C++ that accurately simulates the behavior of this text editor based on user inputs.

Considerations:

- *The text editor begins with an empty document, ready to accept user input.*
- *INSERT operation: Users can add text to the end of the current document, with each insertion limited to 20 English letters to maintain readability and consistency.*
- BACKSPACE operation: Users can erase the last character of the document, provided it's not already empty.
- UNDO operation: Users can revert the effects of the last successful INSERT or BACKSPACE operation, ensuring they can easily correct mistakes or backtrack through their editing history.
- Each operation should only be considered successful if it results in a tangible change to the document.
- It's essential to prevent users from undoing an operation more than once to maintain the integrity of the editing process.
- **You need to create a simulation system that continuously prompts the user for the next operation to perform.**

Example: Consider a user composing a message using the smart keyboard application:

1. User inputs "INSERT Hello": The document now contains "Hello".

2. User inputs "INSERT World!": The document now reads "HelloWorld!".
3. User inputs "BACKSPACE": The exclamation mark is removed, leaving "HelloWorld".
4. User inputs "UNDO": The previous BACKSPACE operation is reversed, restoring the exclamation mark at the end of the document.
5. User inputs "UNDO" again: Since the last action was an UNDO operation, nothing changes in the document.
6. User inputs "BACKSPACE" twice: The last two characters "ld" are removed, resulting in "HelloWor".
7. User inputs "INSERT d": The letter 'd' is appended to the document, producing the final text: "HelloWord".

Note: You must create your classes and cannot use built-in libraries like stack and queue for this task.

Question 2: AVL Tree Implementation for Employee Management System **[Marks - 60]**

You've been tasked with developing an efficient employee management system for a growing organization. To organize and manage employee data effectively, you're required to implement an AVL tree data structure capable of handling integer values representing employee IDs. Additionally, you need to incorporate advanced functionalities to ensure data consistency and provide valuable insights into employee performance. Your objective is to design and implement an AVL tree data structure to manage integer-based employee IDs within the organization. The AVL tree should support standard operations such as insertion, deletion, and search, while also incorporating additional functionalities to ensure data integrity and provide insightful analytics.

Considerations:

- **Fun insert (10 marks):** Implement insertion functionality to add new employee IDs to the AVL tree while maintaining its balanced structure to optimize search and retrieval operations.
- **Fun remove (10 marks):** Implement deletion functionality to remove existing employee IDs from the AVL tree, ensuring that the tree remains balanced and retains its efficiency.
- **Fun search (5 marks):** Implement search functionality to allow users to quickly retrieve employee IDs.
- **Fun validate (15 marks):** Implement a validation mechanism to traverse each node in the AVL tree and check whether the value of each node is equal to the sum of the values of its immediate left and right child nodes. For NULL values, consider the value to be 0.
- **Fun printExtremeCorners (15 marks):** Incorporate a feature to print the nodes of extreme corners of each level in the AVL tree but in alternate order, providing valuable insights into the organizational hierarchy.