4 DATA STRUCTURES

Lab Exercise # 4: Singly Linked Lists and File I/O

Objective

In this lab assignment, you will implement various methods for a **DoublyLinkedList** class. The goal is to understand and apply concepts related to doubly linked lists, including adding and removing elements, handling edge cases, and implementing algorithms for list manipulation.

Instructions

Download the starter code file named **lab4.cpp** and complete the definitions for the following methods in the **DoublyLinkedList** class.

Tasks

1. Constructor **(0.5 Point)**

DoublyLinkedList(): Implement a constructor that initializes an empty doubly linked list by creating **head** and **tail** sentinel (dummy) nodes. The head node should point to the tail node, and vice versa.

2. Destructor **(1 Point)**

~DoublyLinkedList(): Implement a destructor to clean up all dynamically allocated nodes, including the sentinel nodes, to prevent memory leaks.

3. Check if List is Empty **(0.5 Points)**

bool empty() const: Return true if the list is empty (i.e., contains only sentinel nodes) and false otherwise.

4. Retrieve First Element **(0.5 Points)**

const std::string& front() const: Return the element stored in the first valid node of the list. Throw an exception if the list is empty.

5. Retrieve Last Element **(0.5 Points)**

const std::string& back() const: Return the element stored in the last valid node of the list. Throw an exception if the list is empty.

6. Insert at Front **(0.5 Points)**

void addFront(const std::string& elem): Insert a new node containing elem at the front of the list (after the head sentinel node).

7. Insert at Back **(0.5 Points)**

void addBack(const std::string& elem): Insert a new node containing elem at the back of the list (before the tail sentinel node).

8. Remove from Front **(1 Point)**

void removeFront(): Remove the first valid node from the list. Throw an exception if the list is already empty.

9. Remove from Back **(1 Point)**

void removeBack(): Remove the last valid node from the list. Throw an exception if the list is empty.

10. Check if List is a Palindrome **(1 Points)**

bool isPalindrome() const: Implement a function that returns true if the linked list is a palindrome and false otherwise. A palindrome is a sequence that reads the same forward and backward, e.g.,

- 1 – 2 – 3 – 2 – 1
- 1 – 2 – 2 – 1
- 3

  This method **must throw an exception** if the list is empty. You **cannot** use an auxiliary or temporary list.

11. Display List Elements (Already Provided in the starter-code)

void display() const: This method displays all elements of the list. (Already included in the starter code.)

12. Reverse the List **(2 Points)**

void reverseList(): Implement a function to reverse the linked list. Throw an exception if the list is empty.

Additional Requirements

Error Handling

- Exceptions must be derived from the C++ exception class (e.g., runtime_error, logic_error). The catch block in main() should be able to catch these exceptions properly.

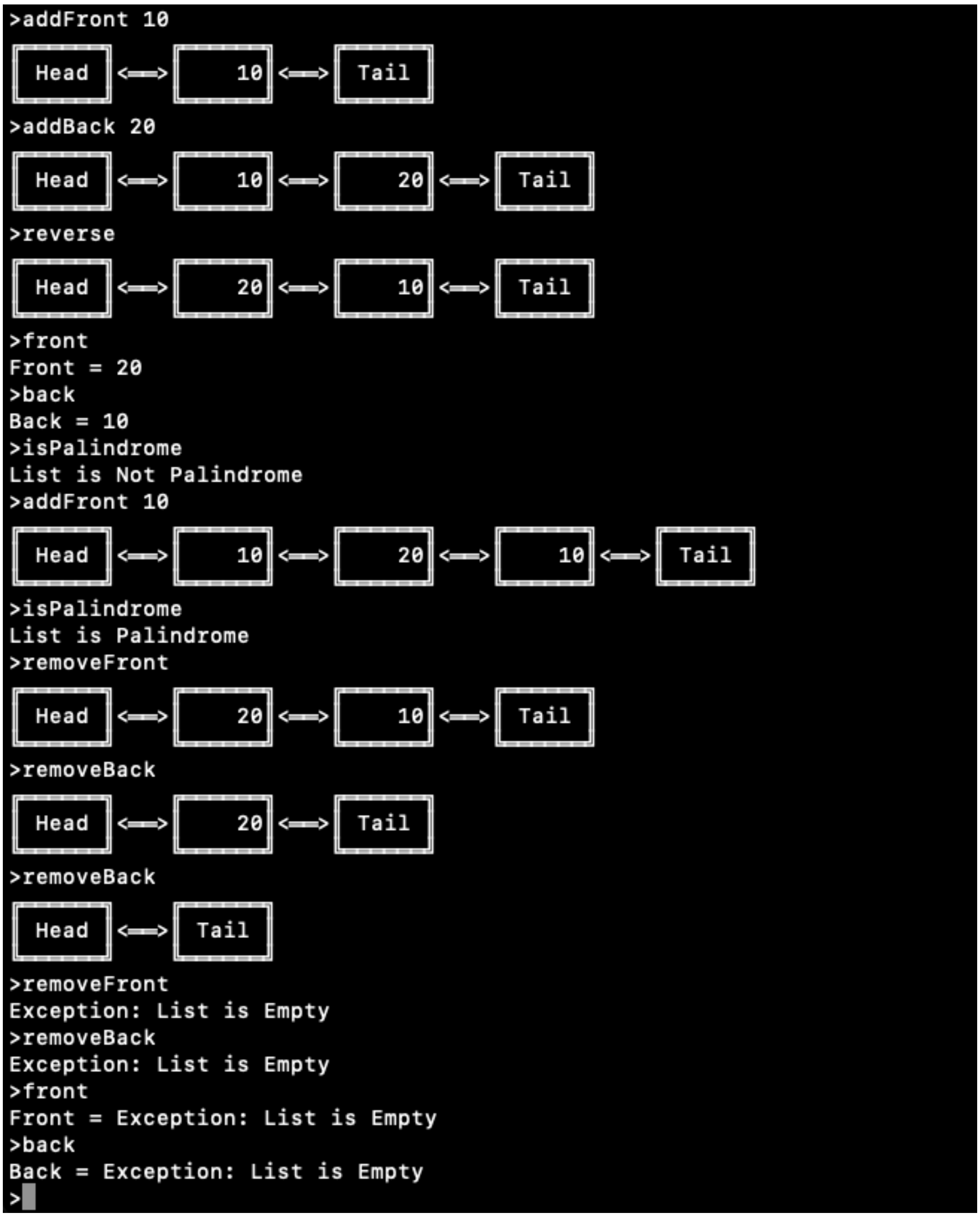
Code Comments **(1 Points)**

- Your code should include **meaningful comments** explaining each method's functionality, edge cases, and key operations.

Submission Guidelines

- Submit the completed **lab4.cpp** file.
- Ensure proper formatting, indentation, and comments.
- Test your code thoroughly before submission.

Sample output



```
>addFront 10
╔══════╗     ╔══════╗     ╔══════╗
║ Head ║<═══>║    10║<═══>║ Tail ║
╚══════╝     ╚══════╝     ╚══════╝
>addBack 20
╔══════╗     ╔══════╗     ╔══════╗     ╔══════╗
║ Head ║<═══>║    10║<═══>║    20║<═══>║ Tail ║
╚══════╝     ╚══════╝     ╚══════╝     ╚══════╝
>reverse
╔══════╗     ╔══════╗     ╔══════╗     ╔══════╗
║ Head ║<═══>║    20║<═══>║    10║<═══>║ Tail ║
╚══════╝     ╚══════╝     ╚══════╝     ╚══════╝
>front
Front = 20
>back
Back = 10
>isPalindrome
List is Not Palindrome
>addFront 10
╔══════╗     ╔══════╗     ╔══════╗     ╔══════╗     ╔══════╗
║ Head ║<═══>║    10║<═══>║    20║<═══>║    10║<═══>║ Tail ║
╚══════╝     ╚══════╝     ╚══════╝     ╚══════╝     ╚══════╝
>isPalindrome
List is Palindrome
>removeFront
╔══════╗     ╔══════╗     ╔══════╗     ╔══════╗
║ Head ║<═══>║    20║<═══>║    10║<═══>║ Tail ║
╚══════╝     ╚══════╝     ╚══════╝     ╚══════╝
>removeBack
╔══════╗     ╔══════╗     ╔══════╗
║ Head ║<═══>║    20║<═══>║ Tail ║
╚══════╝     ╚══════╝     ╚══════╝
>removeBack
╔══════╗     ╔══════╗
║ Head ║<═══>║ Tail ║
╚══════╝     ╚══════╝
>removeFront
Exception: List is Empty
>removeBack
Exception: List is Empty
>front
Front = Exception: List is Empty
>back
Back = Exception: List is Empty
>
```

CODE OF CONDUCT

All assignments are graded, meaning we expect you to adhere to the academic integrity standards of NYU Abu Dhabi. To avoid any confusion regarding this, we will briefly state what is and isn't allowed when working on an assignment/lab-task.

Any documents and program code that you submit must be fully written by yourself. You can, of course, discuss your ideas with fellow students, as long as these discussions are restricted to general solution techniques. Put differently, these discussions should not be about concrete code you are writing, nor about specific results you wish to submit. When discussing an assignment with others, this should never lead to you possessing the complete or partial solution of others, regardless of whether the solution is in paper or digital form, and independent of who made the solution, meaning you are also not allowed to possess solutions by someone from a different year or course, by someone from another university, or code from the Internet, etc. This also implies that there is never a valid reason to share your code with fellow students, and that there is no valid reason to publish your code online in any form.

Every student is responsible for the work they submit. If there is any doubt during the grading about whether a student created the assignment themselves (e.g. if the solution matches that of others), we reserve the option to let the student explain why this is the case. In case doubts remain, or we decide to directly escalate the issue, the suspected violations will be reported to the academic administration according to the policies of NYU Abu Dhabi.

https://students.nyuad.nyu.edu/campus-life/community-standards/policies/academic-integrity/