CL2001 – Data Structure Lab

Exercise # 05

Note:

- Copied task will be awarded **zero** marks.
- Use comments wherever applicable.
- Submit a pdf file containing all your C++ code with all possible screenshots of every task output on Google Classroom. The name of file should be your roll no followed by your name (roll-no-name.pdf) i.e., (23P-1234-Ali.pdf).
- Variables and functions names should be meaningful.

Problem: 1 | Doubly Linked List

You are tasked with creating a robust inventory management system for a retail store using a doubly linked list. Develop a menu-driven C++ program to facilitate the following operations:

1. **Add Product**: Allow the user to input details for a new product, including its unique ID, name, price, and quantity. Ensure that products are inserted into the inventory in ascending order of their IDs.
2. **Remove Product**: Enable the user to delete a product from the inventory by providing its ID. Ensure that the list remains sorted after removal.
3. **Display Products**: Display all products in the inventory sorted by their IDs. Include their IDs, names, prices, and quantities.
4. **Update Price**: Allow the user to update the price of a product by entering its ID and the new price.
5. **Find Product**: Implement a feature to search for a product by its ID. Display the product's name, price, and quantity if found; otherwise, notify the user that the product does not exist.

Ensure that the program handles edge cases effectively, such as attempting to remove or update a non-existent product record. Utilize clear user prompts and error messages to enhance usability.

Problem: 2 | Split Doubly Linked List

Create two doubly linked lists out of one doubly linked list such that 1st linked list contains even data and 2nd linked list contains odd data of the provided linked list.

Input:

NULL <– 1 ⇔ 2 ⇔ 3 ⇔ 4 ⇔ 5 ⇔ 6 ⇔ 7 ⇔ 8 ⇔9 –>NULL

Output:

1st: NULL <– 2 ⇔ 4 ⇔ 6 ⇔ 8->NULL

2nd: NULL <– 1 ⇔ 3 ⇔ 5 ⇔ 7 ⇔ 9->NULL

Problem: 3 | Circular Linked List

You've been tasked with developing a menu-driven C++ program to manage a playlist using a circular linked list. Implement the following functionalities:

1. **Add Song at Beginning**: Allow the user to input details for a new song and insert it at the beginning of the playlist.
2. **Remove Song at End**: Enable the user to delete the last song from the playlist.
3. **Display Playlist**: Display all songs in the playlist.

Ensure that the program handles the circular nature of the linked list correctly, allowing seamless insertion and deletion of songs at the beginning and end of the playlist.