CSE 4304-Data Structures Lab. Winter 2022-23
Date: 29 Aug 2023
Target Group: SWE B
Topic: Linked lists

Instructions:

- Regardless of when you finish the tasks in the lab, you have to submit the solutions in the Google Classroom. The deadline will always be at 11.59 PM of the day in which the lab has taken place.
- Task naming format: <fullID>_<Task><Lab><Group>.c/cpp. Example: 170041034_T01L02A.cpp
- If you find any issues in the problem description/test cases, comment in the google classroom.
- If you find any test case that is tricky that I didn't include but others might forget to handle, please comment! I'll be happy to add.
- Use appropriate comments in your code. This will help you to easily recall the solution in the future.
- Obtained marks will vary based on the efficiency of the solution.
- Do not use the <bits/stdc++.h> library.

Task-01:

Implement the basic operations using a Linked list. Your program should include the following functions:

1. **Insert_front**(int key):
    - Insert the element with the 'key' at the beginning of the list.
    - Time Complexity: O(1)
2. **Insert_back**(int key):
    - Insert the element with the 'key' at the end of the list.
    - Time Complexity: O(1)
3. **Insert_after_node** (int key, int v):
    - Insert a node with the 'key' after the node containing the value 'v' if it exists. (shows error message otherwise).
    - Time complexity: $O(n)$
4. **Update_node** (int key, int v):
    - Looks for the node with value *v* and updates it with the new value 'key' (error message if the node doesn't exist)
    - Time complexity: O(n)
5. **Remove_head** ():
    - Remove the first node from the linked list.
    - Time complexity: O(1)
6. **Remove_element** (int key):
    - Removes the node containing the 'key' if it exists (else throw an error message).
    - Time complexity O(n)
7. **Remove_end** ():
    - Remove the last node from the linked list.
    - Time complexity: O(n)

Input format:

- The program will offer the user the following operations (as long as the user doesn't use option 7):
    - Press 1 to insert at front
    - Press 2 to insert at back
    - Press 3 to insert after a node
    - Press 4 to update a node
    - Press 5 to remove the first node
    - Press 6 to remove a node
    - Press 7 to remove the last node
    - Press 8 to exit.
- After the user chooses an operation, the program takes necessary actions (or asks for further info if required).

Output format:

- After each operation, the status of the list is printed.

Task 02

- Satisfy the requirements of Task-1 using **'Doubly linked list'**.
- One additional requirement is, that after each operation, you have to print the linked list twice:
  - From head to tail.
  - From the tail towards the head (don't use recursive implementation, rather utilize the 'previous' pointers).
- The **Remove_end()** function should be done in **O(1)**

Task 03

Find the attached code Task 03.cpp and complete the function:

```
ListNode *findMiddleNode(ListNode *head)
```

Note: You can only write code inside the function. Keep the rest of the things as it is.

Task 04

A set of sorted numbers is stored in a linked list. Your task is to keep the first occurrence of a number and remove the other duplicate values from the list.
(Stop taking input when the user enters -1)

Input	**Output**
2 7 7 10 12 18 25 25 25 27 -1	2 7 10 12 18 25 27
5 5 5 5 5 -1	5
1 2 3 4 5 -1	1 2 3 4 5
10 20 20 20 20 20 20 -1	10 20

Note: Your solution should remove the node from the existing linked list instead of using a new linked list to store unique elements.