CSE 4304-Data Structures Lab. Winter 2021
Lab-03
Date: June 05, 2021 (Saturday)
Target Group: All Lab groups
Topic: Linked Lists

Instructions:

- Task naming format: fullID_T01L04_2A.c/cpp
- If you find any issues in problem description/test cases, comment in the google classroom.
- If you find any tricky test case which I didn't include and others might forget to handle, please comment! I'll be happy to add.
- Modified sections will be marked with BLUE colour.

Task 1

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 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_element** (int key):
    - Removes the node containing the 'key' if it exists (else throw an error message).
    - Time complexity O(n)
6. **Remove_end** ():
    - Remove the last node from the linked list.
    - Time complexity: O(1)

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 a node
  - Press 6 to remove the last node
  - Press 7 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 2

- Satisfy the requirements of Task-1 using ‘Doubly linked list’.
- One additional requirement is, after each operation, you have to print the linked list twice:
  - From head towards the tail.
  - From tail towards the head (don’t use recursive implementation, rather utilize the ‘previous’ pointers).

Task 3

Implement a ‘Deque’ using ‘Double linked list’. Your program should offer the user the following options:

1. void **push_front**(int key): Inserts an element at the beginning of the list.
2. void **push_back**(int key): Inserts an element at the end of the list.
3. int **pop_front**(): Extracts the first element from the list.
4. int **pop_back**(): Extracts the last element from the list.

Note:

- The maximum time complexity for any operations is O(1).
- For option 3,4: the program shows an error message if the list is empty.

Input format:

- The program will offer the user the following operations (as long as the user doesn’t use option 5):
  - Press 1 to push_front
  - Press 2 to push_back
  - Press 3 to pop_front
  - Press 4 to pop_back
  - Press 5 to exit.
- After the user chooses an operation, the program takes necessary actions (or asks for further values if required).

Output format:

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

Task 4

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.

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