Data Structures and Algorithms

Lab 03 – Linked Lists

Exercises/Tasks:

1. Write a program that implements all the methods of a singly linked list, as mentioned below:
    - **addToFront**: Adds a new node with the given data to the front of the linked list.
    - **getFrontItem**: Returns the data stored in the first node (front) of the linked list.
    - **removeFrontItem**: Removes the first node (front) from the linked list.
    - **addToBack**: Adds a new node with the given data to the end (back) of the linked list.
    - **getBackItem**: Returns the data stored in the last node (back) of the linked list.
    - **removeBackItem**: Removes the last node (back) from the linked list.
    - **find**: Checks if the given key is present in the linked list.
    - **Remove**: Removes the node with the given key from the linked list, if present.
    - **isListEmpty**: Checks if the linked list is empty.
    - **addKeyBeforeNode**: Adds a new node with the given key before the node containing the specified data in the linked list.
    - **addKeyAfterNode**: Adds a new node with the given key after the node containing the specified data in the linked list.
    - **printAll**: Prints all the values in the linked list.

    Also, implement the main method to show/test how the different operations are performed on the list.

2. Extend the singly linked list mentioned in the question above by adding the "**tail**" to it. Then, update the methods **addToBack** and **removeBackItem** to see if the efficiency increases.
3. In linked lists, you cannot directly access elements using indices as you can with arrays. Write a program to simulate index-based operations (insertion, deletion, access) by traversing the linked list until you reach the desired position/index. Thus, update the linked list created in Task 1 above and add three methods (**get(index)**, **insertAt(index, data)**, **removeFrom(index)**) to perform the respective operations when called by the user and provided an index (like done in arrays). Also, implement

a method named **getSize** or **getLength** to count and return the total number of elements in the linked list.

4. Implement a method **reverse(head)** to reverse the singly linked list in place (in place means you will not use an extra linked list but you can use some extra (pointer) variables).