**Homework 2: Playing with LinkedList**

You have been provided three different linked list implementations: and . The files implement following versions of linked list:

* is a singly linked list implementation with a head pointer.
* is a singly linked list with an additional tail pointer.

For this homework, you are required to add different functionalities to the provided implementation.

**Task 1 – 4 are to be added in LinkedList.cpp file.**

The file implements a singly linked list using a head pointer that points to the first node of the list. Your job is to add following features.

***Task 1: Add insertLast function.*** Add a new function to the list. This function will insert a new item at the end of the list.

***Task 2: Add insertAfter function.*** Add a new function to the list. This function will insert a new item after an existing item in the list. The function will first search for in the list. Then it will insert the after the in the list. If the is not present in the list, then insertion should be discarded.

***Task 3: Add deleteFirst function.*** This function will delete the first element of the list. You must ensure that memory of the deleted item is released properly. In case the item is not found in the list, return a NULL\_VALUE, otherwise return SUCCESS\_VALUE.

***Task 4: Add deleteLast function.*** This function will delete the last element of the list. You must ensure that memory of the deleted item is correctly released. In case the item is not found in the list, return a NULL\_VALUE, otherwise return SUCCESS\_VALUE.

**Task 5 - 6 is to be added in LinkedListWithTail.cpp file.**

The file includes an additional pointer that should always point to the last node of the list. In the given implementation, pointer is not set properly in implemented functions: and . Your job is to add following features.

***Task 5: Set pointer correctly.*** Add required codes in and functions to set up the pointer correctly so that it always points to the last node of the linked list.

***Task 6: Make function efficient.*** For this task, your job is to use this pointer to make the function more efficient than your first implementation in Task 1 above, which runs in time. If you have a pointer, then you will not require searching the whole list to find the end of the list. So, change your previous implementation accordingly so that new version runs in constant time. Ensure that the pointer is correctly set after insertion.

**You must also satisfy the following requirements:**

* You must extend the given code.
* You cannot use any function of C library except and input output functions.
* You should avoid object-oriented programming.
* You must *free* unused memory where it is required.
* ***You must not use other’s code. You must not share your code. You must not copy from any other sources such as web, friends, relatives, etc. In all cases, you will earn a 0 and will move closer to getting an “F” grade in the course.***