Lab 06
Data Structures
BS DS Fall 2024 Morning/Afternoon

Instructions:

- **Attempt the following tasks exactly in the given order.**
- Make sure that there are no ***dangling pointers*** or ***memory leaks*** in your programs.
- Indent your code properly.
- Use meaningful variable and function names. Follow the naming conventions.
- Use meaningful prompt lines/labels for all input/output that is done by your programs.

Task # 0 *(Pre-Requisite)* *(Max Time: 15 Minutes)*

Implement a **singly linked list** class which stores integers in **unsorted** order. Your class declarations should look like:

```
class Node {
      int data;
      Node* next;
      Node();
      Node(int d);
};
```

```
class LinkedList {    private:
      Node* head;
  public:
      LinkedList();
       ~LinkedList();
 // Insertion functions
    void insertAtHead(T val);
    void insertAtTail(T val);

    // Deletion functions
    void removeAtHead();
    void removeAtTail();
    void remove(T val);

    // Utility functions
    bool search(T key);
    void update(T key, T val);
    int countNodes();
       …
};
```

Apart from the **default constructor** and **destructor**, the `LinkedList` class should also have the following public member functions:

`void display ()`

This function displays the contents of the linked list on screen.

Task # 1 *(Max Time: 20 Minutes)*

Now, add the following three public member functions to the **`LinkedList`** class:

`bool removeKthNode (int k, int& val)`

This function will remove the **k**th element (node) from the linked list. For example, if the linked list object **`list`** contains **{4 2 8 1 9 5 4 6}**, then the function call **`list.removeKthNode(4)`** should remove the 4th element (node) from the linked list and the resulting **`list`** should be: **{4 2 8 9 5 4 6}**. Before deallocating the node, this function should store the data present in that node in the variable **`val`**. This function should return false if the linked list contains fewer than **k** elements; otherwise it should remove the **k**th node from the linked list and return true. (*Note:* You are NOT allowed to modify the data of any node in the linked list).

Also write a drive main function to test the working of each of the above-mentioned functions.

Task # 2 *(Max Time: 20 Minutes)*

Add the following public member function to the **`LinkedList`** class:

`void combine (LinkedList& list1, LinkedList& list2)`

This function will combine the nodes of the two linked lists (**`list1`** and **`list2`**) to form one list. All the nodes of the first list (**`list1`**) will precede all the nodes of the second list (**`list2`**).

For example, if **`list1`** contain **{7 3 4 2}** and **`list2`** contains **{5 9}**, then after the function call **`list3.combine(list1,list2)`**, **`list3`** should contain **{7 3 4 2 5 9}** and **`list1`** and **`list2`** should be empty now.

Note: You are NOT allowed to create any new node in this function. You are also NOT allowed to modify the "data" field of any node. You can assume that the LinkedList object on which this function is called is empty at the start of this function.

Also write a driver main function to test the working of the above function.

Task # 3 *(Max Time: 25 Minutes)*

Add the following public member function to the **`LinkedList`** class:

`void shuffleMerge (LinkedList& list1, LinkedList& list2)`

This function will merge the nodes of the two linked lists (**`list1`** and **`list2`**) to make one list, by taking nodes alternately from the two lists.

For example, if **`list1`** contains **{2 6 4}** and **`list2`** contains **{8 1 3}**, then after the function call **`list3.shuffleMerge(list1,list2)`**, **`list3`** should contain **{2 8 6 1 4 3}** and **`list1`** and **`list2`** should be empty now.

Note: You are NOT allowed to create any new node in this function. You are also NOT allowed to modify the "data" field of any node. You can assume that both lists (which are being merged) have the **same number of elements**. You can also assume that the LinkedList object on which this function is called is empty at the start of this function.

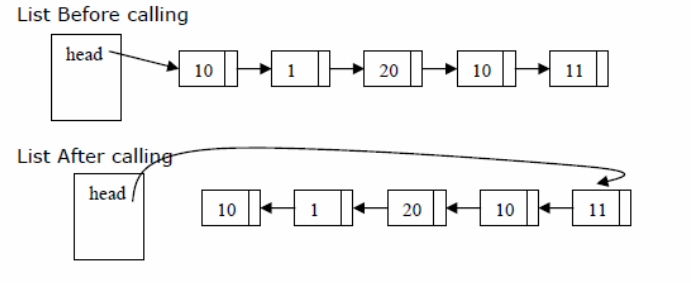
Also write a driver main function to test the working of the above function.

Task # 4 *(Max Time: 15 Minutes)*

Reverse the link list structure iteratively

```
void reverseList()
```

Very Important: *You are NOT allowed to create any new node in this function. You are also NOT allowed to modify the "info" field of any node.*



Task # 5 *(Max Time: 25 Minutes)*

Remove all the duplicate nodes in a link list

```
void removeDuplicates()
```

When this function will be called on some link list then after the execution of this function the list will not contain any node with its info repeating more then once. E.g. if list contains 23, 5, 4, 23, 6, 78, 4, 5 then after the list will contain only 23, 5, 4, 6, 78