Lab 3

DSA BS DS Fall 2022

Task 1

Complete the following ADT of Link List as discussed in the lectures.

```
class Node:
    def __init__(self, val=None):
        self.info = val
        self.next = None

class LinkList:
    def __init__(self):
        self.head = None

    def insert_at_head(self, val):
    def insert_at_tail(self, val):
    def insert_after(self, key, val):
    def insert_before(self, key, val):
    def search(self, key):
    def display(self):
    def update(self, key, val):
    def remove_at_head(self):
    def remove_at_tail(self):
    def remove_after(self, val):
    def remove_before(self, val):
    def remove(self, val):

# Main function
if __name__ == "__main__":
    obj = LinkList()
    obj.insert_at_head(2)
    obj.insert_at_head(3)
    obj.insert_at_tail(9)
    obj.insert_after(3,4)
    obj.insert_before(9,8)

    obj.display()
```

Task 2

Remove KthNode

Implement the following public member function of the **`LinkedList`** class:

```
def removeKthNode(self, k):
```

This function will remove the **k**th element (node) from the linked list.

For example, if the linked list object **`list`** contains these 8 values **{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 contain these 7 values: **{4 2 8 9 5 4 6}**.

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 info of any node in the linked list).

Task 3

Combine Lists

```
def combine(self, list1, list2):
```

This function should combine the nodes of the two linked lists (**list1** and **list2**) into one list. All the nodes of the first list (**list1**) will precede (come before) all the nodes of the second list (**list2**).
For example, if **list1** contain **{7 3 4 2}**, **list2** contains **{5 9}**, and **list3** is Empty, 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.

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. You can* ***assume*** *that the LinkedList object on which this function is called is empty at the start of this function.*

Hint 1: Do a lot of **paperwork** before writing the code. Also, make sure that all the **boundary cases / special cases** are properly handled.

Task 4

Shuffle Merge

Implement the following public member function of the **LinkedList** class:

```
def shuffleMerge(self, list1, list2):
```

This function should shuffle-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}**, **list2** contains **{8 1 3}**, and **list3** is Empty, 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.
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. You can assume that both lists (which are being shuffle-merged) contain the* ***same number of elements/nodes****. You can also assume that the LinkedList object on which this function is called is empty at the start of this function.*
Hint: Do a lot of **paperwork** before writing the code. Also, make sure that all the **boundary cases / special cases** are properly handled.

Task 5

Remove all the duplicate nodes in a link list

```
def removeDuplicates(self):
```

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

Task 6

Reverse the link list structure iteratively and recursively

```
def reverseList(self):
```

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.*