Worksheet:
Linked Lists

Recap

- Learn to reverse a Linked List!
- A Linked List is a chain of Nodes. A Node is composed of a Value and a Pointer to the next Node (usually called next)
- Use a dummy node when you're not sure what the Head of the LinkedList will be
- Most of the time, you'll have to use multiple pointers to solve these questions
- Useful trick: To detect cycle, have one pointer move twice as fast as the other pointer
- Doubly Linked Lists are Linked Lists that have two pointers, one pointing to the next node and one pointing to its previous node.

Exercise 1

Learn the basics of Linked List questions

The following questions are rarely asked because they might be too easy, but this will help you build the foundation for Linked Lists. Post them on the FB group if you want to get feedback on your code.

Reverse a singly linked list.

Example:

```
Input: 1->2->3->4->5->NULL
Output: 5->4->3->2->1->NULL
```

Follow up:

A linked list can be reversed either iteratively or recursively. Could you implement both?

Remove Linked List Elements

Example:

```
Input:  1->2->6->3->4->5->6, val = 6
Output: 1->2->3->4->5
```

Odd Even Linked List

Given a singly linked list, group all odd nodes together followed by the even nodes. Please note here we are talking about the node number and not the value in the nodes.

You should try to do it in place. The program should run in O(1) space complexity and O(nodes) time complexity.

Example 1:

```
Input: 1->2->3->4->5->NULL
Output: 1->3->5->2->4->NULL
```

Example 2:

```
Input: 2->1->3->5->6->4->7->NULL
Output: 2->3->6->7->1->5->4->NULL
```

Note:

- The relative order inside both the even and odd groups should remain as it was in the input.
- The first node is considered odd, the second node even and so on ...

Palindrome Linked List

Given a singly linked list, determine if it is a palindrome.

Example 1:

```
Input: 1->2
Output: false
```

Example 2:

```
Input: 1->2->2->1
```

```
Output: true
```

Follow up:

Could you do it in O(n) time and O(1) space?

Exercise 2

Understanding Doubly Linked Lists

To fully understand Doubly Linked Lists, the best way to do it is to make one yourself. Feel free to use any language, but here is a skeleton you can use in Python3:

```
class MyLinkedList:

    def __init__(self):
        """
        Initialize your data structure here.
        """


    def get(self, index: int) -> int:
        """
        Get the value of the index-th node in the linked list. If the index
is invalid, return -1.
        """


    def addAtHead(self, val: int) -> None:
        """
        Add a node of value val before the first element of the linked
list. After the insertion, the new node will be the first node of the
linked list.
        """


    def addAtTail(self, val: int) -> None:
        """
        Append a node of value val to the last element of the linked list.
        """
```

```
    def addAtIndex(self, index: int, val: int) -> None:
        """
        Add a node of value val before the index-th node in the linked
list. If index equals to the length of linked list, the node will be
appended to the end of linked list. If index is greater than the length,
the node will not be inserted.
        """


    def deleteAtIndex(self, index: int) -> None:
        """
        Delete the index-th node in the linked list, if the index is valid.
        """



# Your MyLinkedList object will be instantiated and called as such:
# obj = MyLinkedList()
# param_1 = obj.get(index)
# obj.addAtHead(val)
# obj.addAtTail(val)
# obj.addAtIndex(index,val)
# obj.deleteAtIndex(index)
```