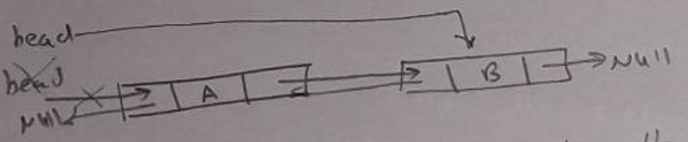
Deletion

1] at the beginning



* make the head points to the next of the first node, Then reset the prev of the next to Null

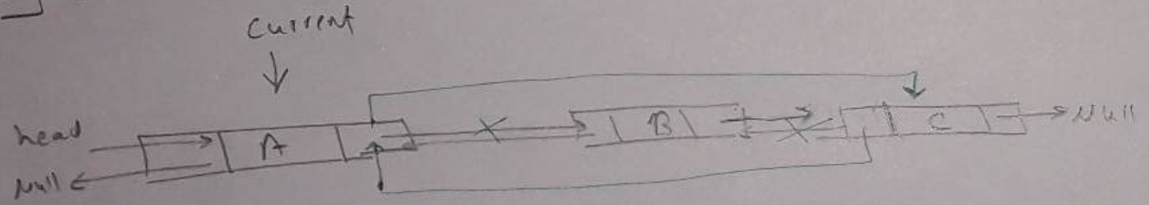
→ current = head

→ head = Current → next , Current = Current → next

→ Current → next = Null

Time O(1), space O(1)

2] at the Mid after a given node



* make the next of the given node jump twice

i.e Current → next = Current → next → next

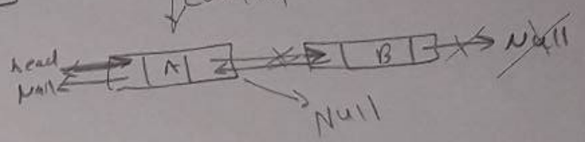
* Move the current one step forward so it is at C

Current = Current → next

* set previous to A Current → previous = Current → previous → previous

Time O(1), space O(1)

3] Delete at the end



* Traverse till Current → next = Null

* put next of the current to Null

Then deallocate B

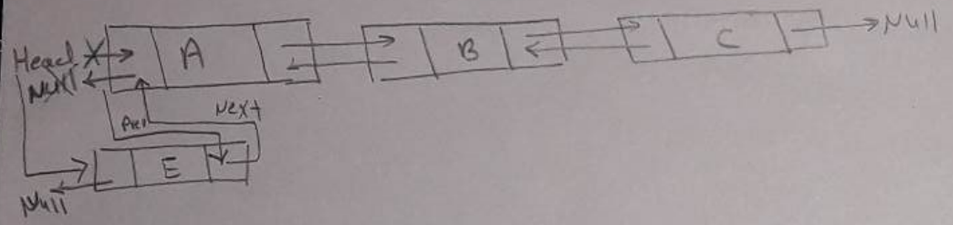
O(n) Time, O(1) space

Comparison:	Start	Mid	End
Insertion:			
consider Traversal	easiest	hardest	easy
Not consider Traversal	easy	hardest	easiest

Deletion	Start	Mid	End
	easiest (need No Auxillary)	hardest	easy

insertion in D L L

1] At the beginning of a Linked List

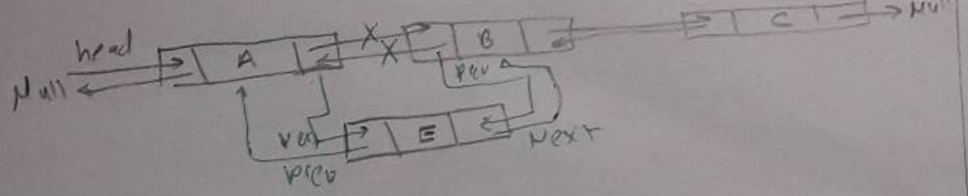


Steps:
* Create the new node
* put the Next of the new node to the head
* put the head = the new node

Time complexity: O(1)
since we need no traversal

Auxspace : O(1)

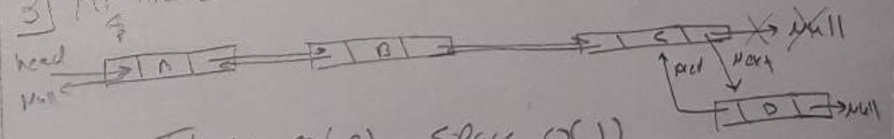
2] insertion at mid i.e. after a given node



Steps:
* Create the New Node
* put the next of the new node in the next of the given node
i.e if given node is A
then newnode → next = current → next
* put the previous of B to the new node
current → next → prev = newnode.
* make the next of A the new node
current → next = newnode
new node → prev = current

Time O(1) since The node is given
space O(1)

3] At the End



Steps Time: O(n) Space O(1)

* Create new node
* Traverse the whole L.L till Current → Next == Null
* make Current → Next = new node
make newnode → prev = current
make newnode → next = Null