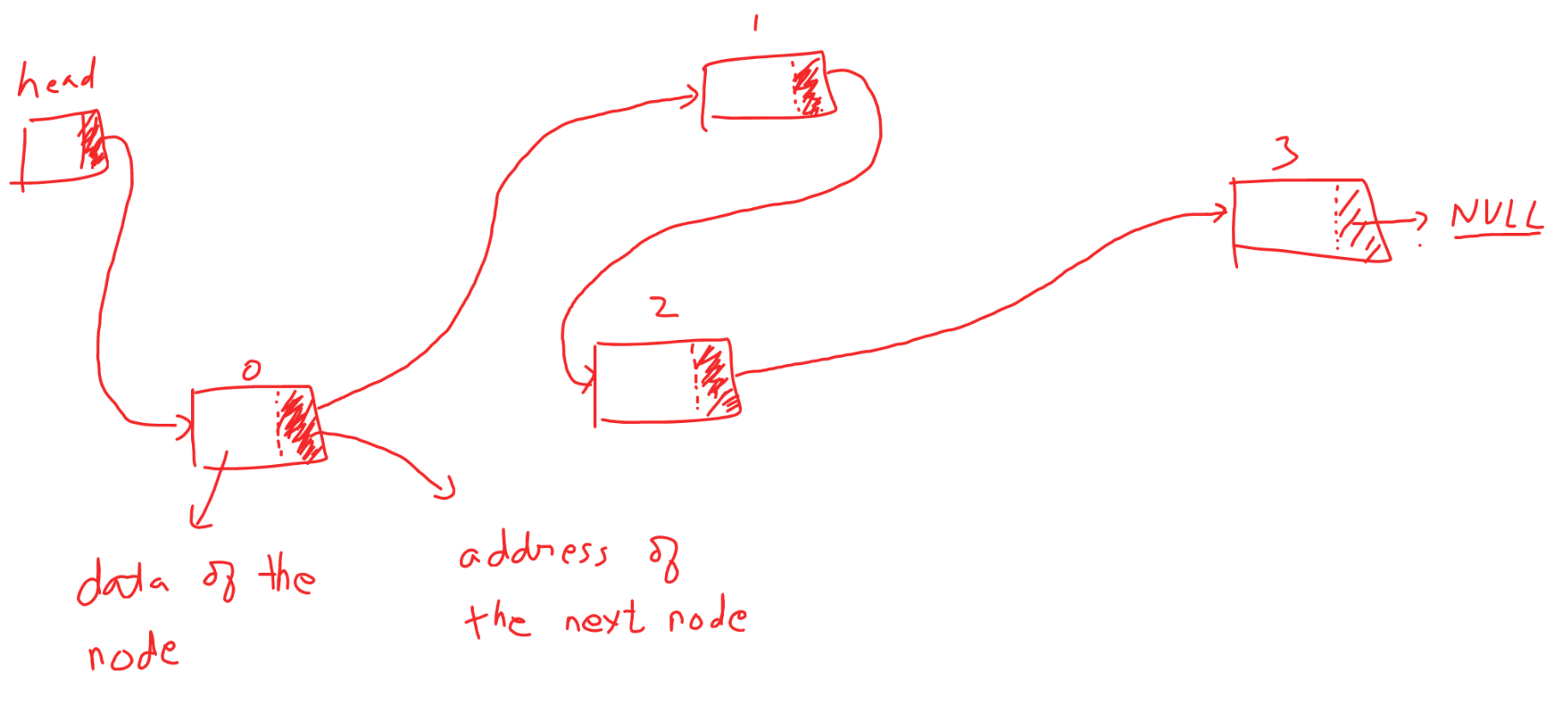
LINKED LIST

⇒ It is a linear-data structure just like an array.
But, the difference is, that LL does not store data is contiguous memory location.
The data is scattered over the memory.


head
0
1
2
3
NULL
data of the node
address of the next node

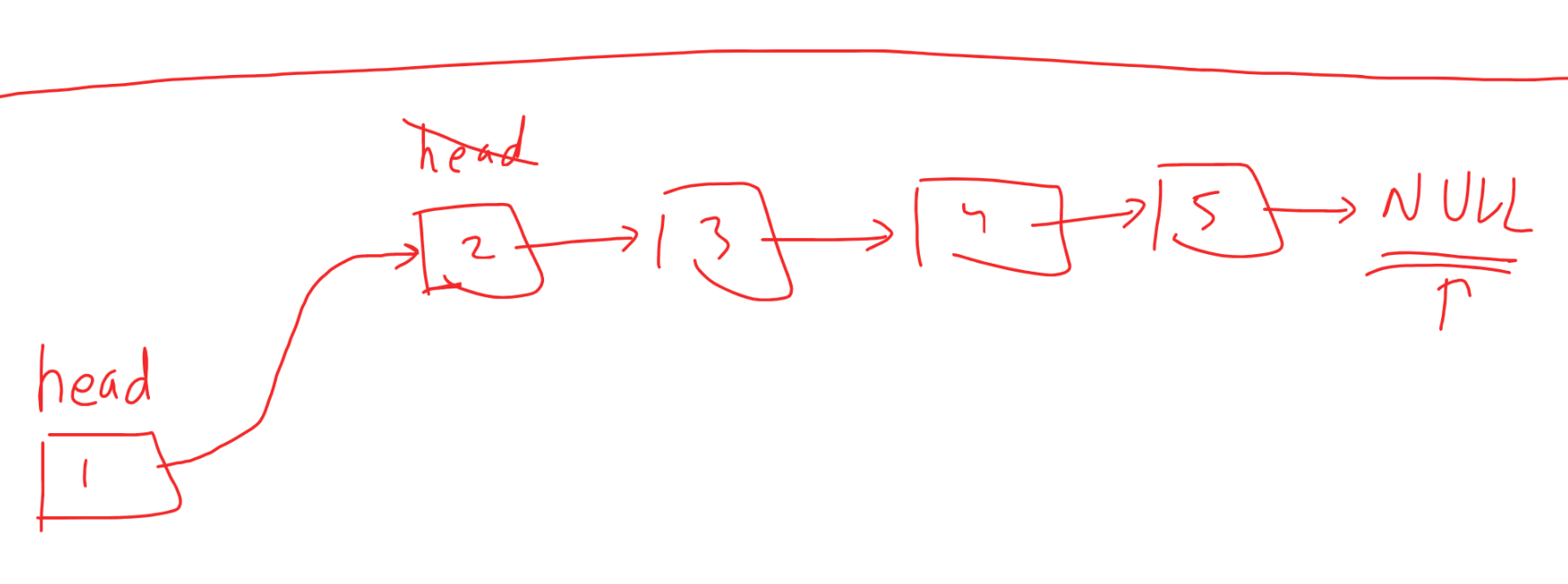
Dynamically resizable arrays:

o/p	size = 0
Push :	size = 1
Push :	size = 2
Push :	size = 4
Push :	size = 4
Push :	size = 8

Remaining space for 3 integers is wasted.

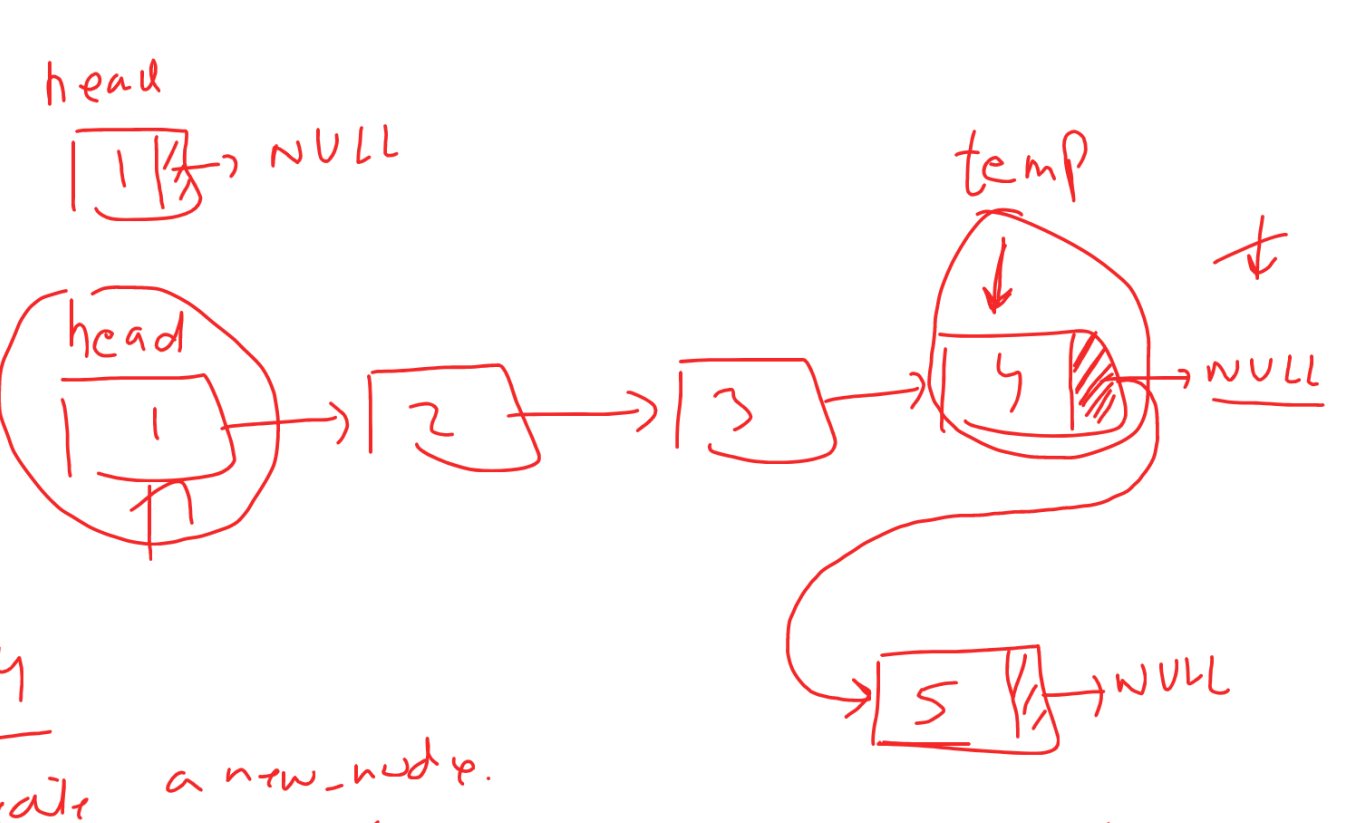
Adv. of LL over arrays:
- → low memory wastage as it does not allocate memory for undeclared nodes.
- → Insertion/deletion at the beginning is O(1).
- → Insertion/deletion from the middle requires no shifting.

Disadvantages of LL over arrays:
- → Random access takes O(n) time in the worst case.
- → Require extra memory for each node (to store the pointer to the next node)
- → LL is not cache-friendly.


~~head~~
head
1
2
3
4
5
NULL

insert at beginning:
- → Create a new_node
- → Make the new_node point to head.
- → Mark new_node as the new head

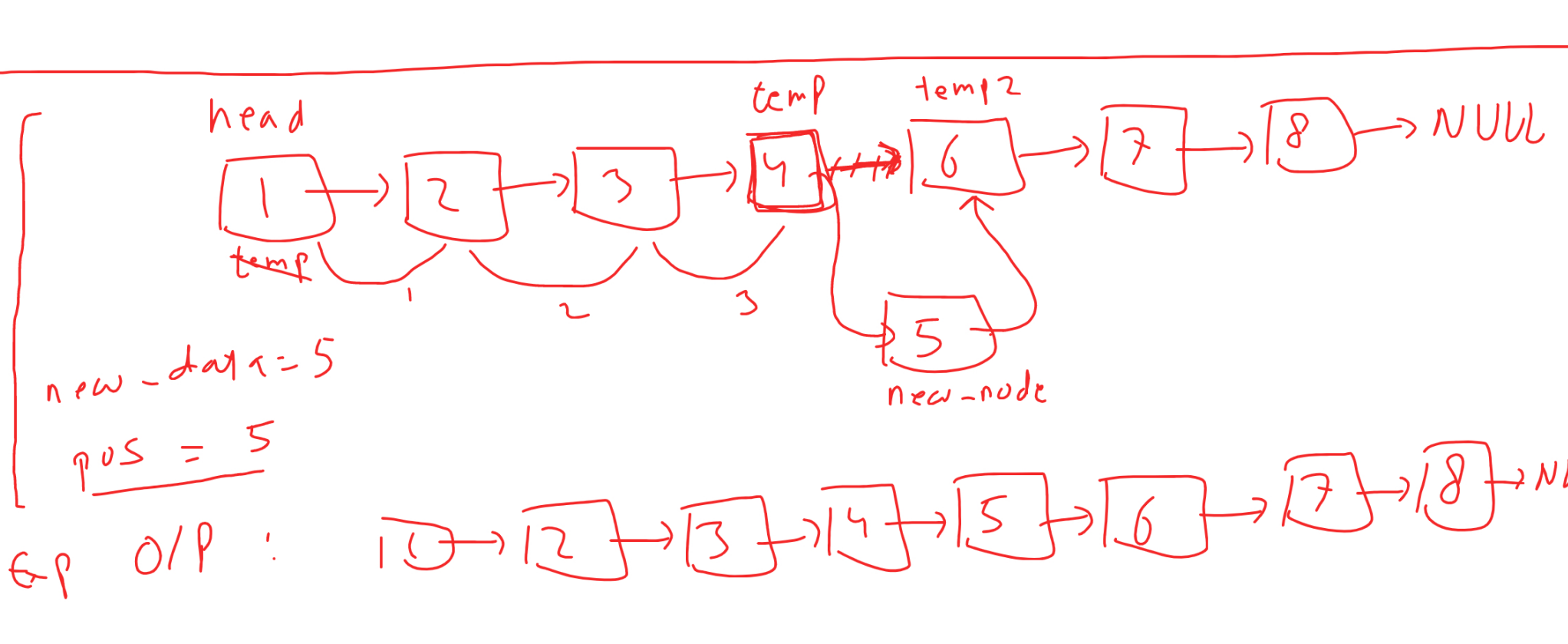
head = ~~NULL~~


head
1
NULL
head
1
2
3
temp
4
NULL
5
NULL

n=4
- → Create a new_node.
- → traverse till (head → next != NULL)
- → Update the 'next' pointer of the last node to point to the new node

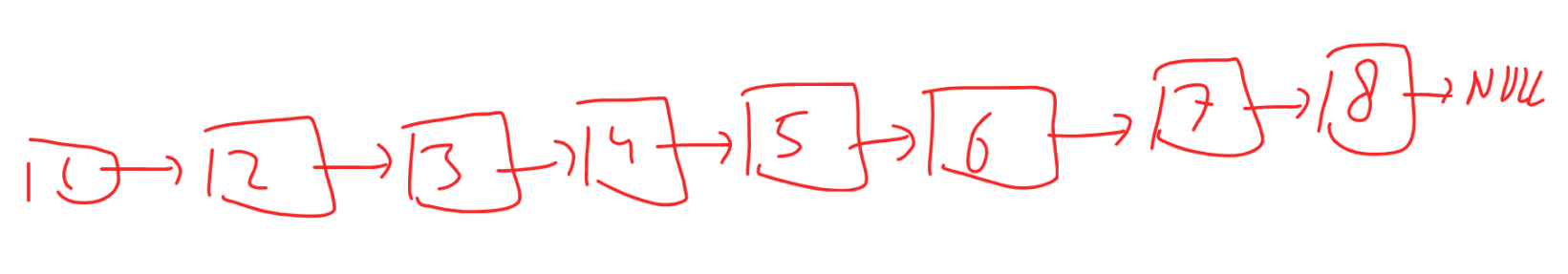
head = NULL

NULL → (next) ⇒ Runtime error (the program will crash)

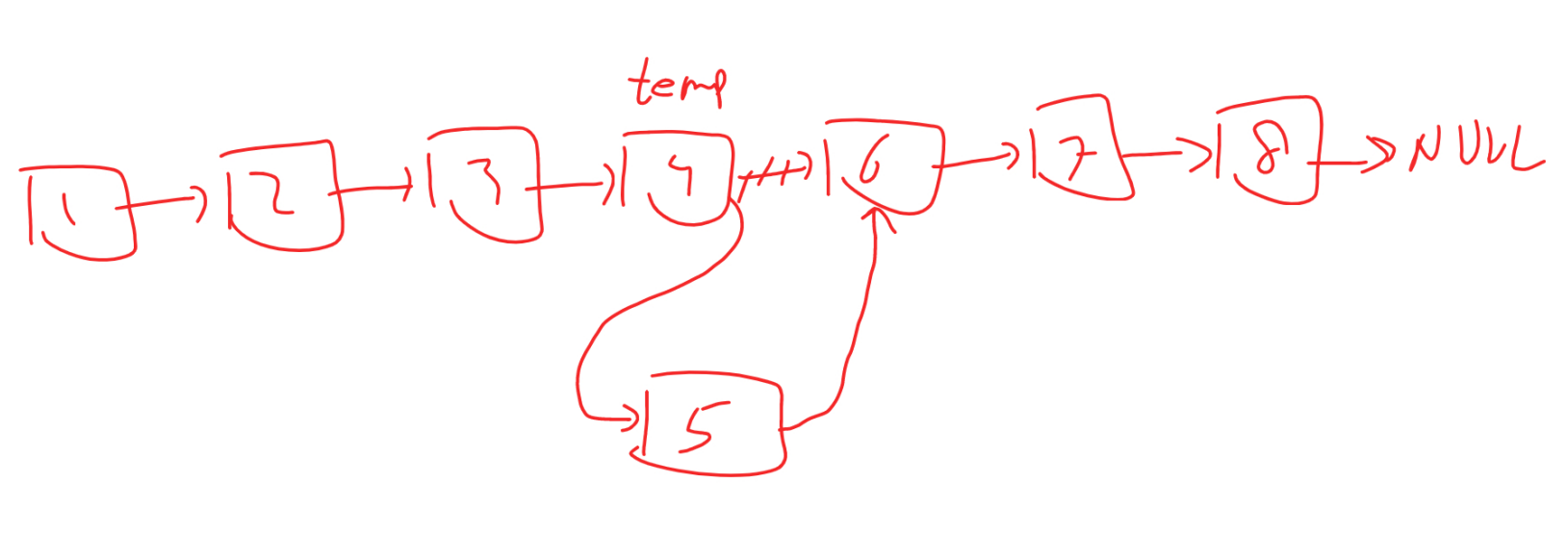

head
1
temp
2
3
temp
4
temp2
6
7
8
NULL
1
2
3
5
new_node

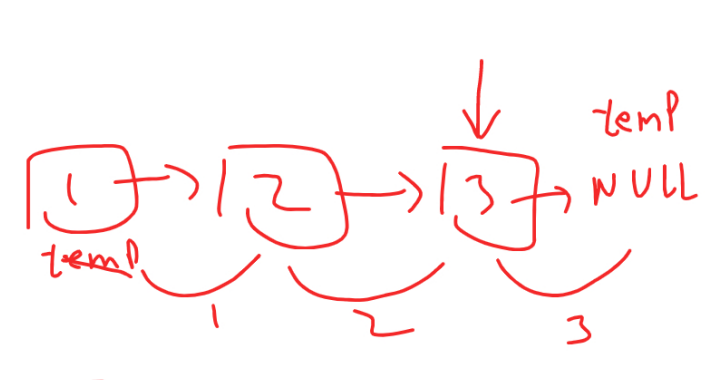
new_data = 5
pos = 5

Gp O/P :


1
2
3
4
5
6
7
8
NULL

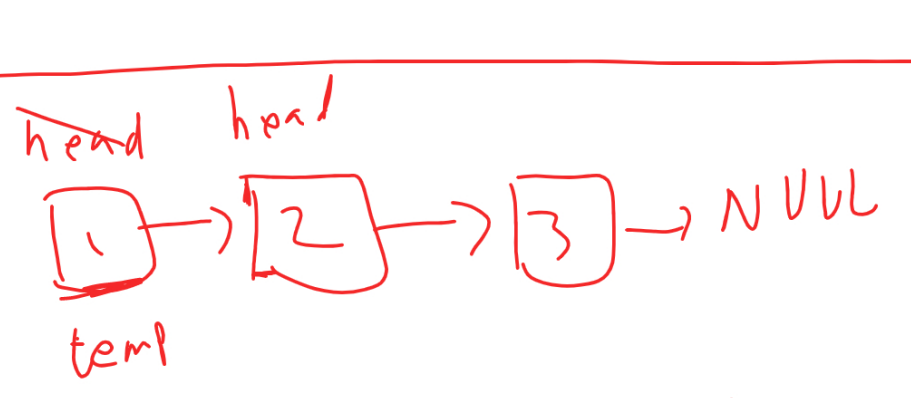
- → temp = head
- → Hop temp by (pos−2) places
- → Create a new_node
- [→ temp2 = temp → next]
- → temp → next = new_node
- → new_node → next = temp2
- → Create new_node
- → new_node → next = temp→next
- → temp → next = new_node


temp
1
2
3
4
6
7
8
NULL
5


1
2
3
temp
NULL
temp
1
2
3

pos : 5

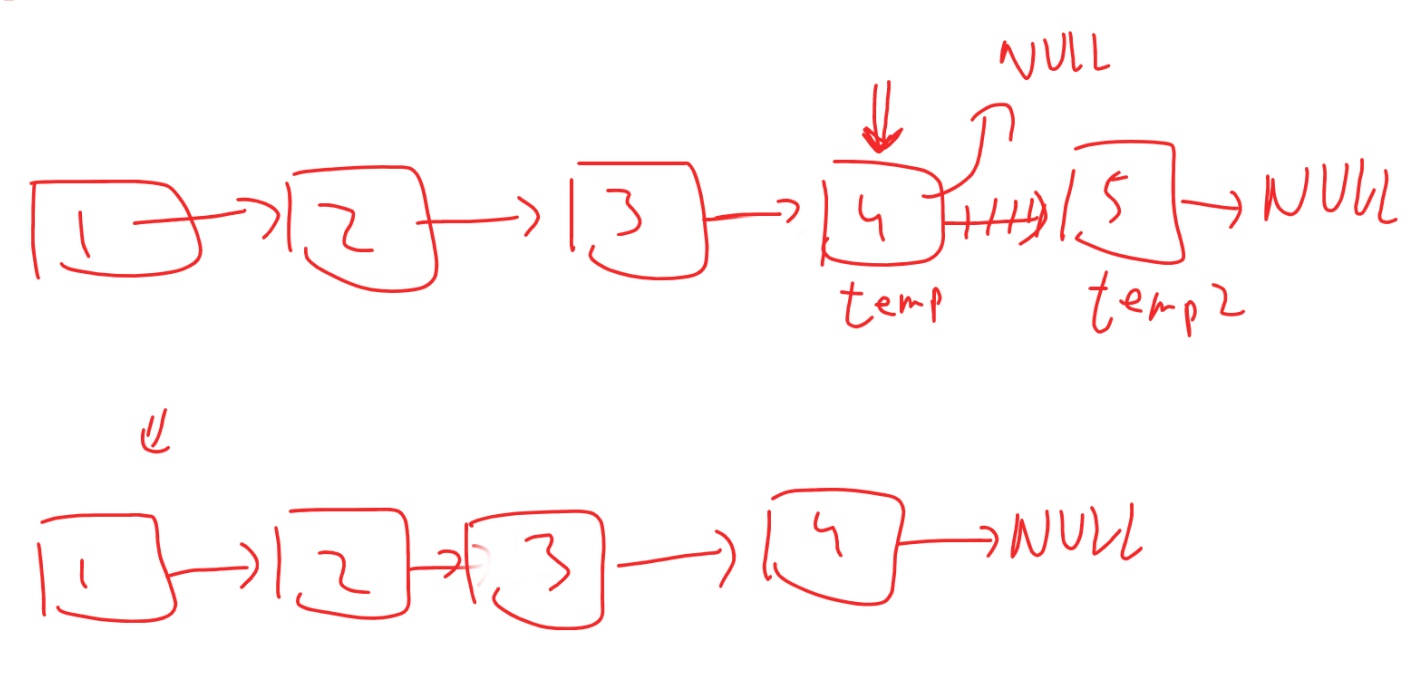
(pos > (len + 1))


~~head~~
head
1
2
3
NULL
temp

2 → 3 → NULL

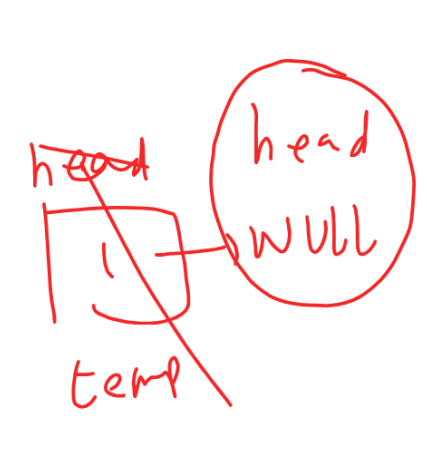
head = head → next

- → temp = head
- → head = head → next
- → delete (temp)

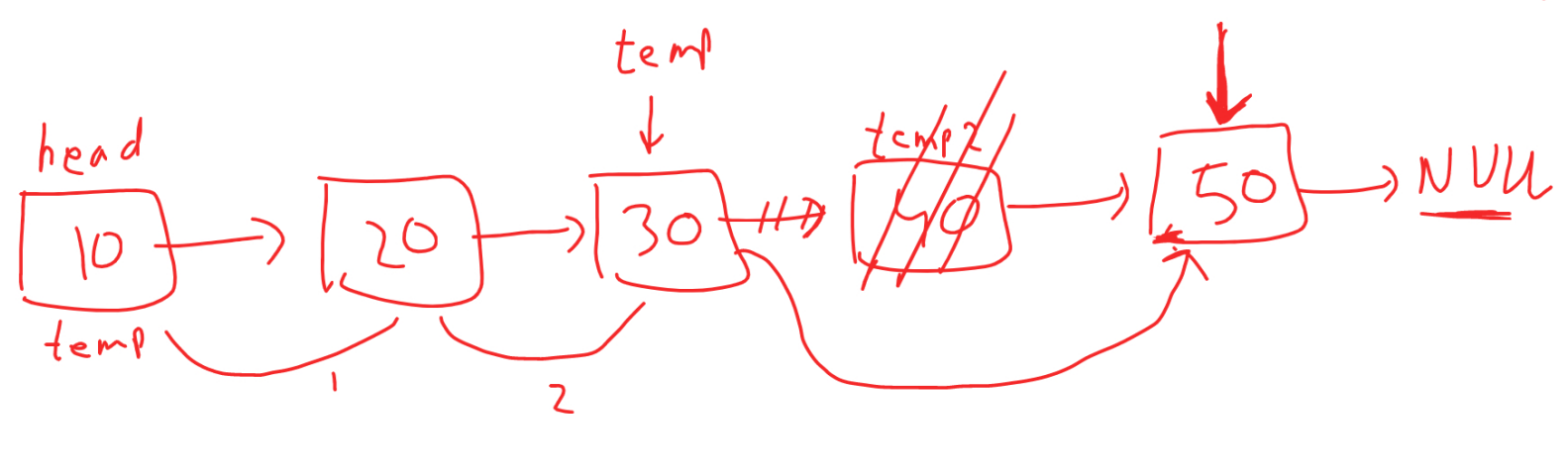

NULL
1
2
3
4
5
NULL
temp
temp2
⇓
1
2
3
4
NULL

- → while (temp → next → next != NULL)
- → temp2 = temp → next
- → temp → next = NULL
- → delete (temp2)

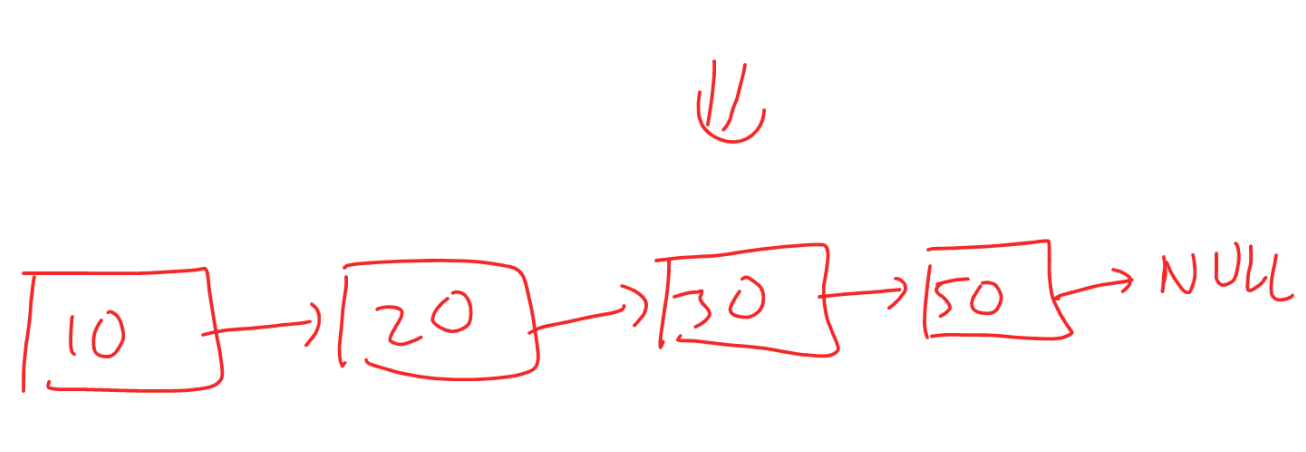
Edge-cases:
1) head == NULL
2) head → next == NULL
] o/p is an empty LL.


~~head~~
head
NULL
1
temp

pos − 2 = 4
pos = 6
pos ≥ 6
⇓
pos > n

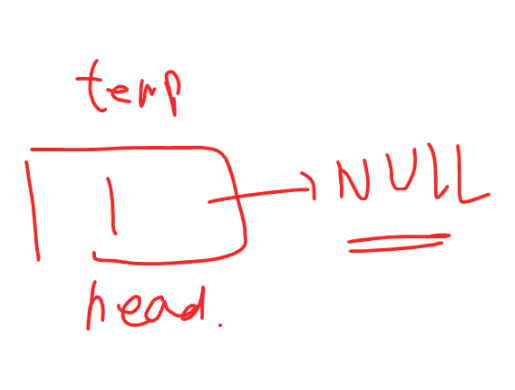

head
10
temp
20
temp
30
temp2
40
50
NULL
1
2

pos = 4


⇓
10
20
30
50
NULL

- → DO (pos − 2) no. of hops of temp = head.
- → temp2 = temp → next
- → temp → next = temp2 → next
- → delete (temp2)

Edge-cases:
1) 0 nodes
2) 1 node
3) pos < 1 or pos > n


temp
1
NULL
head.

pos = 1
↓
−1