Linked List, July - 17

17 July 2020 18:59

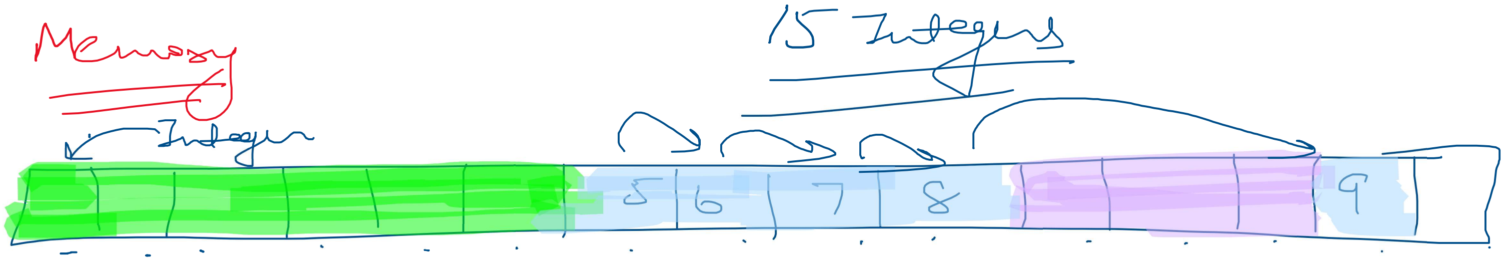
Disadvantages of Array

1. Similar data types
2. Fixed Size
3. Costly Deletion O(n)
4. Costly Insertion(When array is full, or at any particular index)
5. Contiguous Memory Allocation

1	2	3	4	5

1		3	4	5

1	3	4	5	



4 Bytes → 32 bits

A → 6 integers

B → 4 integers

C → 3 integers

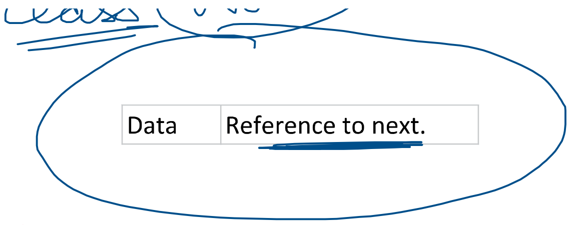
5, 6, 7, 8, 9 ← D → 5 integers

Node

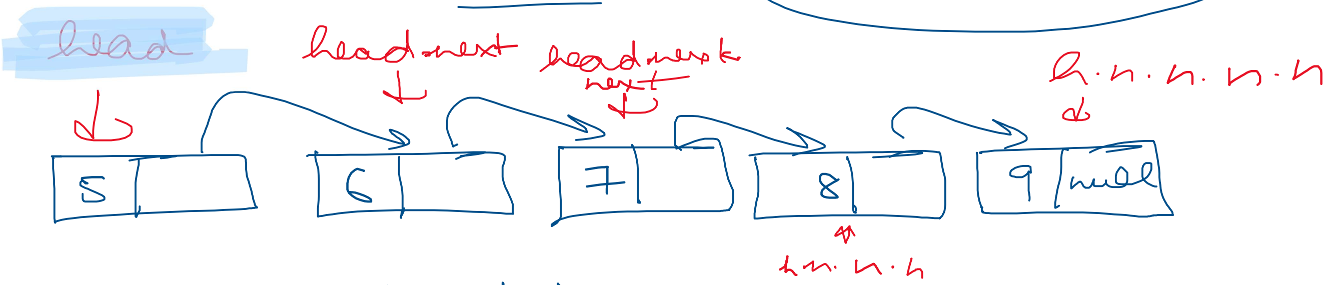
Solution offered:

Store two things:

1. Own Data → data
2. Link to the next place → next



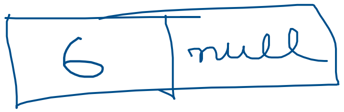
Data	Reference to next.

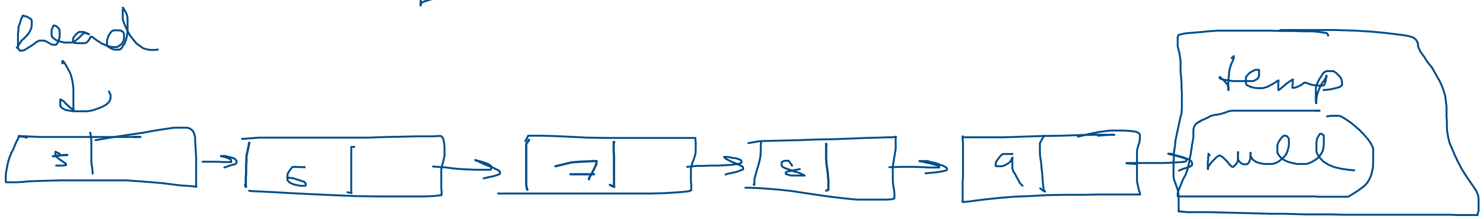


Data → int data

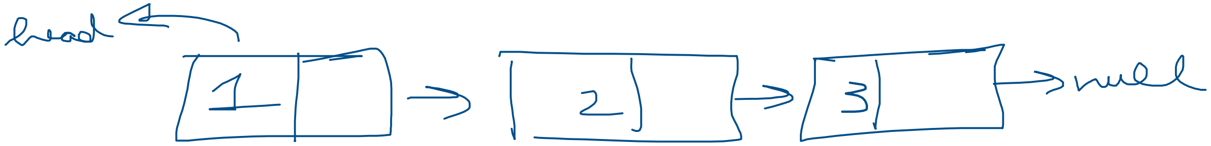
Node next;

```
class Node
{
    int data;
    Node next;
}
```





Insertion

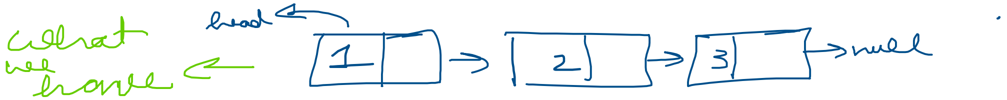


1. Insertion at Head

2 Insertion at End

3 Insertion at nth position.

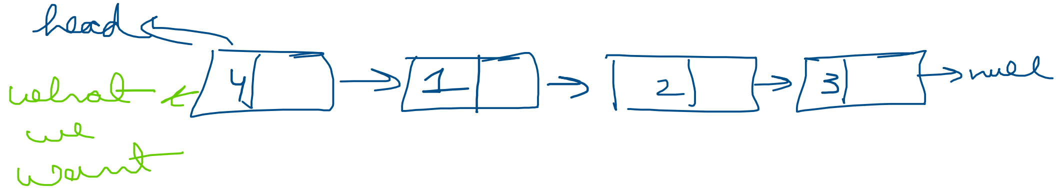
1 Insertion at Head

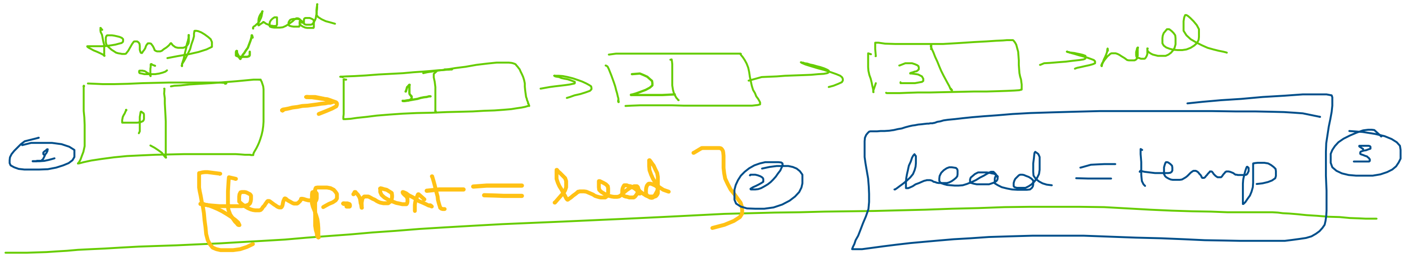


Brahmastra:

1. Draw what you have.
2. Draw what you want.
3. Draw the intermediate steps to reach what you want one by one.

Task InsertAtHead(4)





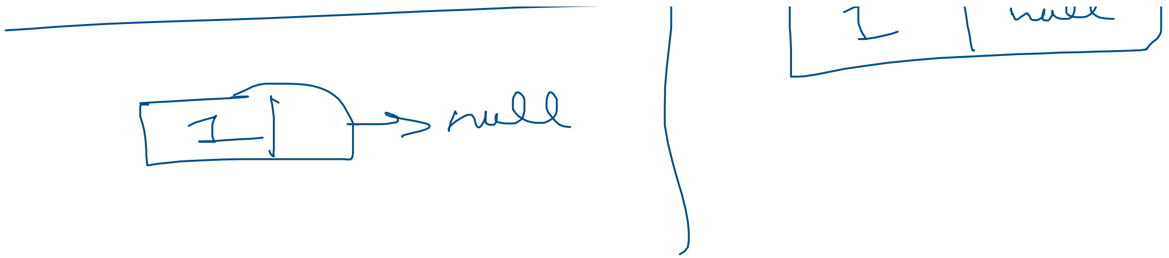
temp.next = head

head = temp

Condition check → If LL is empty.

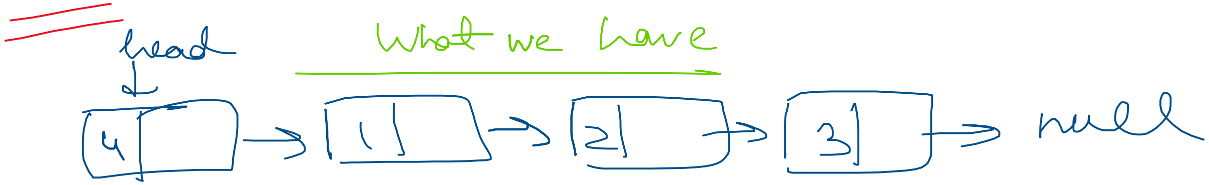
head == null

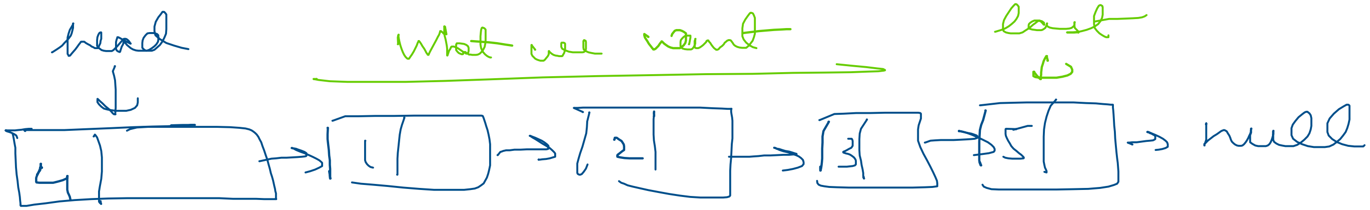
head==null

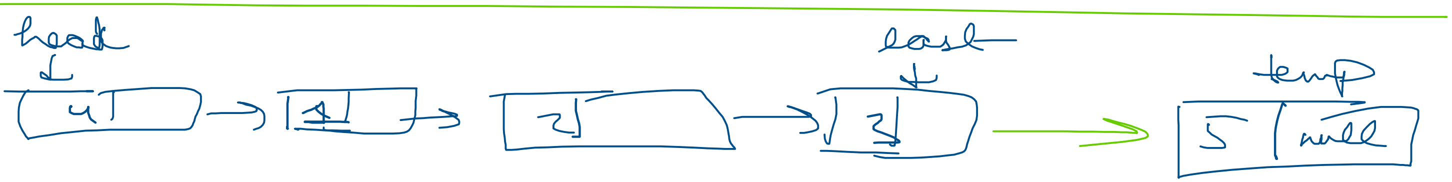


This case has been automatically handled by our code.

2 Insertion at End





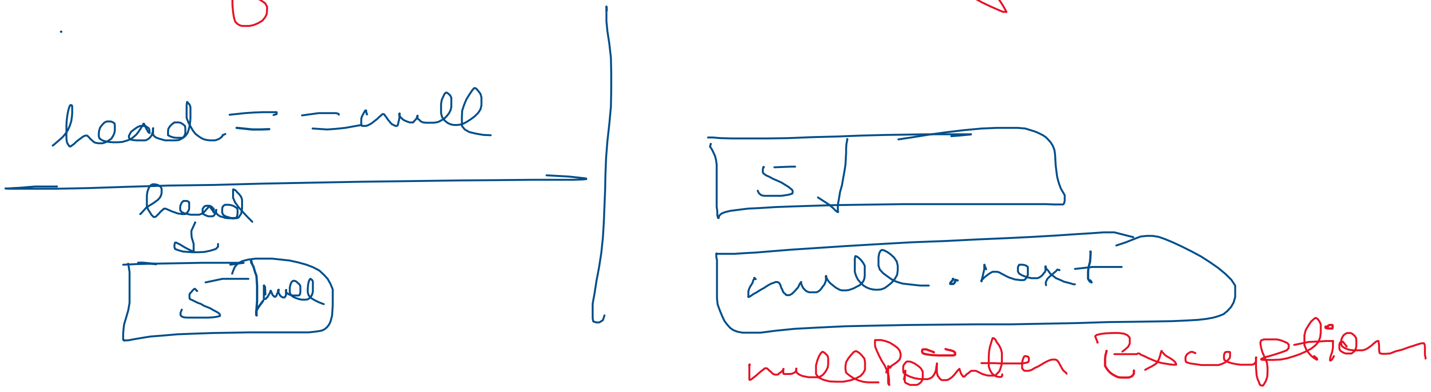


1 => Create a node ✓

2 => Find last Node

3 => last.next = temp ✓

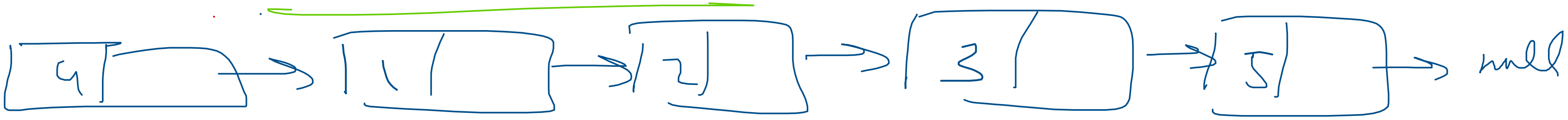
what if Linked List is Empty.



In this case we will have to handle this explicitly

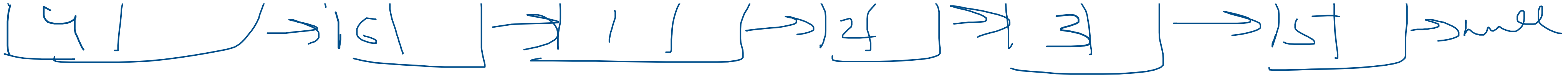
3 Insertion at nth position. H.W

have



insert at Position 2 element 6.

want


4
6
1
2
3
5
null