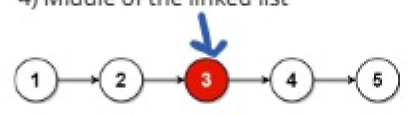
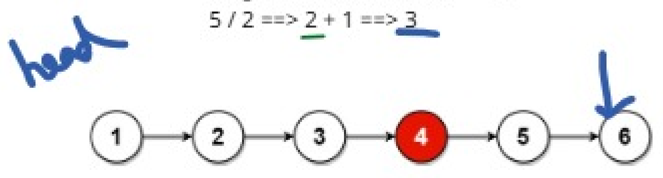
1) Create a singly linked list
2) Insert a node in between
3) Reverse a singly linked list
4) Middle of the linked list



length of the linked list ==> 5
5 / 2 ==> 2 + 1 ==> 3



length = 6
6 / 2 ==> 3 + 1 => 4
count = 0

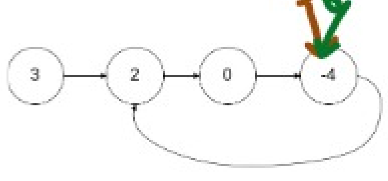
80% ==> linked list problem

slow fast pointer pattern

slow pointer ==> +1
fast pointer ==> +2

head
1 => 2 => 3 => 4 => 5
s f

10 => 15 => 20 => 25 => 30 => 50 null



Linked List Cycle(no end)

true ==> cycle exists
false ==> cycle doesn't exist

Merge 2 sorted linked lists

L1 **10 => 12 => 14 => 25 => 56**

L2 **4 => 6 => 8 => 15 => 100**

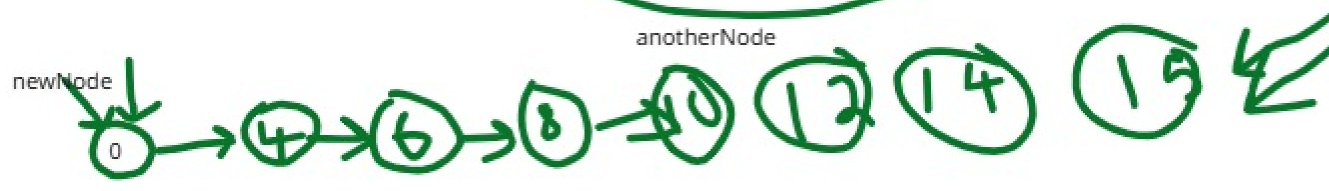
4 => 6 => 8 => 10 => 12 => 14 => 15 ......

h1

L1 **10 => 12 => 14 => 25 => 56**

h2

L2 **4 => 6 => 8 => 15 => 100 ==> 150 ==> 200**

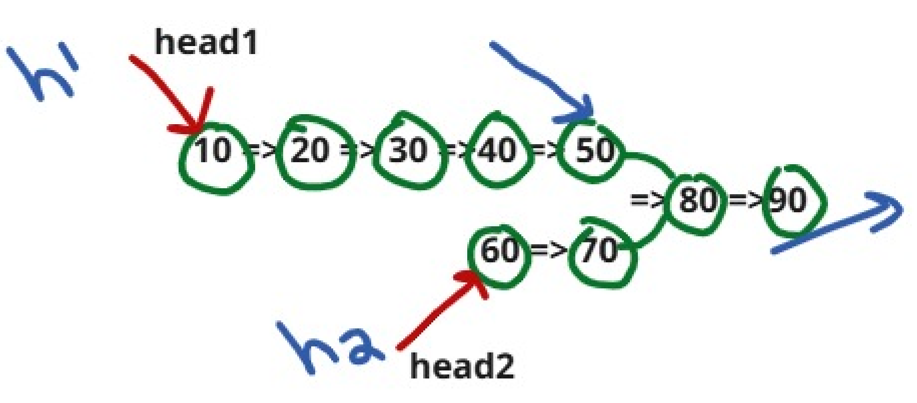


```
ListNode newNode = new ListNode(0)
ListNode anotherNode = newNode;

while(l1 != null && l2 != null)
      if(h1.val < h2.val)
      {

      }
      else if(h2.val < h1.val)
      {

      }
```



10 ==> 60 ==> not equal
10.next ==> 20
60.next ==> 70

20.next => 30
70.next ==> 80

30.next => 40
80.next => 90

h1 → 1st
h2 → 2nd

40.next ==> 50
90.next ==> null

when any of the linked list reaches null, change the path

h1(1st) = 50
h2(2nd) = null

h1 ==> 50
h2 ==> 10
50.next ==> 80
10.next ==> 20

80.next ==> 90
20.next ==> 30

90.next ==> null
30.next ==> 40

h1 = null
h2 = 40

h1 = 60
h2 = 40

60.next ==> 70
40.next ==> 50

70.next ==> 80
50.next ==> 80