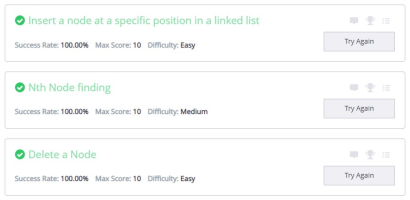


Insert a node at a specific position in a linked list

Success Rate: 100.00% Max Score: 10 Difficulty: Easy

Try Again

Nth Node finding

Success Rate: 100.00% Max Score: 10 Difficulty: Medium

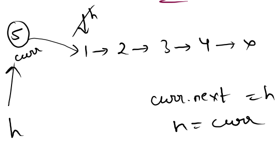
Try Again

Delete a Node

Success Rate: 100.00% Max Score: 10 Difficulty: Easy

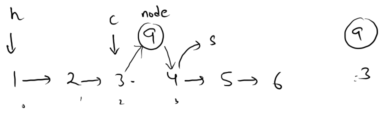
Try Again

1. Insert at Head.



curr.next = h
h = curr

2.

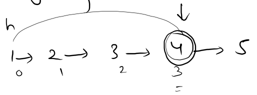


s = c.next.next

c.next = node
node. next = save

3. N^th Node finding

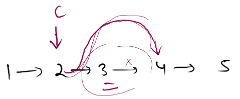
N=2



len = 5

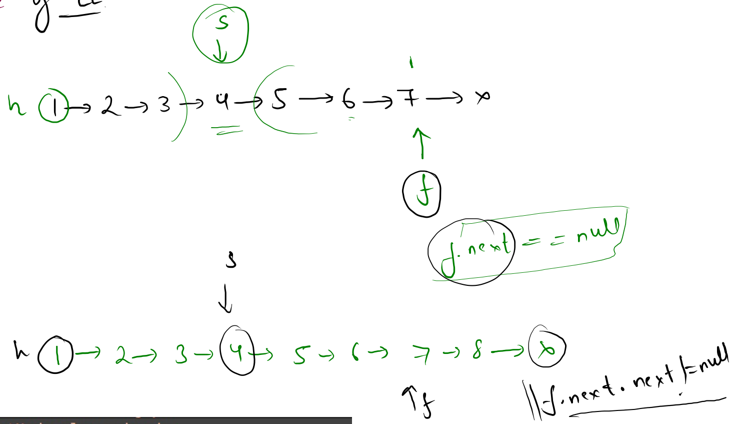
x = 5-2 (len - N)
= 3

4. Delete a node

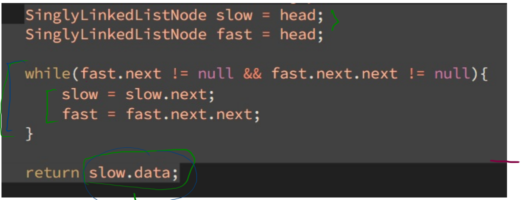


s = c.next
c.next = c.next.next
s.next = null

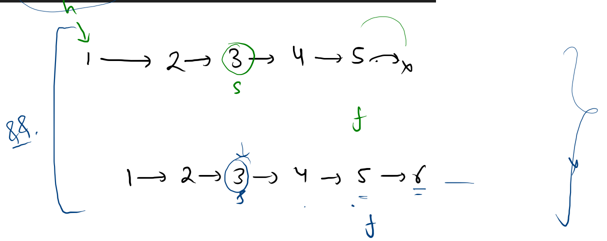
Middle of LL

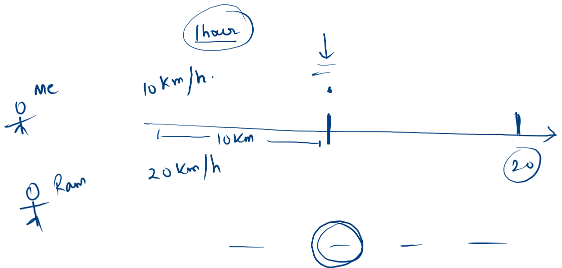


s
h 1 → 2 → 3 → 4 → 5 → 6 → 7 → x
f
f.next == null
s
h 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → x
f
|| f.next.next != null

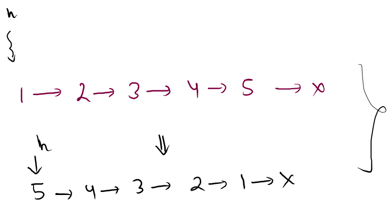


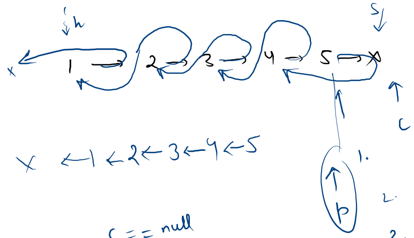
```
SinglyLinkedListNode slow = head;
SinglyLinkedListNode fast = head;

while(fast.next != null && fast.next.next != null){
    slow = slow.next;
    fast = fast.next.next;
}

return slow.data;
```





Reverse a linked list



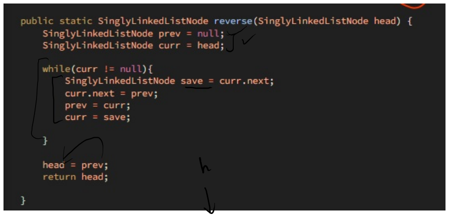


1. s = c.next
2. c.next = p
3. p = c
4. c = s

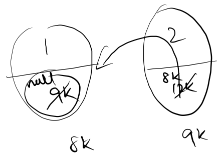
c == null
stop

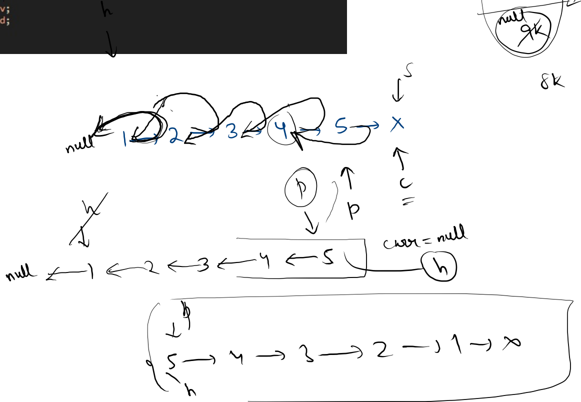
prev = null

$\boxed{h}$ = p

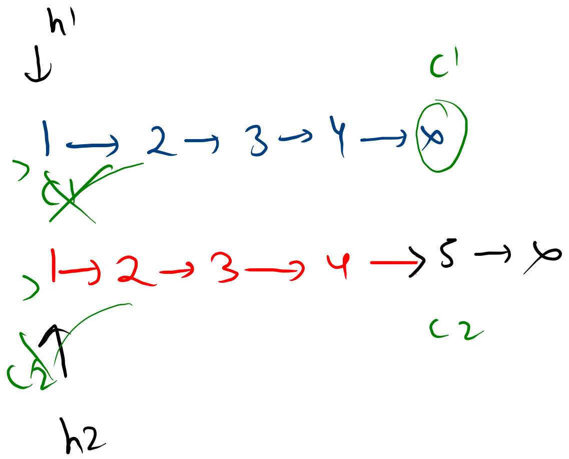


```
public static SinglyLinkedListNode reverse(SinglyLinkedListNode head) {
    SinglyLinkedListNode prev = null;
    SinglyLinkedListNode curr = head;

    while(curr != null){
        SinglyLinkedListNode save = curr.next;
        curr.next = prev;
        prev = curr;
        curr = save;

    }

    head = prev;
    return head;

}
```





Compare two linked lists

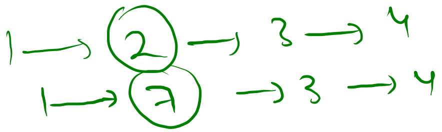


c1.data != c2.data

↳ false

c1 != null || c2 != null

↳ false.



Palindromic Linked List

1 → 2 → 3 → 4 → 5 → x

1 → 2 → 3 → 2 → 1 →

1 → 2 → 3 → | 3 → 2 → 1 → x

logic

mid

nH

L

R

1 → 2 → 3 → 3 → 2 → 1 → x

even

1 → 2 → 3 → x 1 → 2 → 3 → x

Divide (find mid)

Reverse (R)

compareTwoLL(L, R)

→ true

Yes

nH = mid.next

odd

m

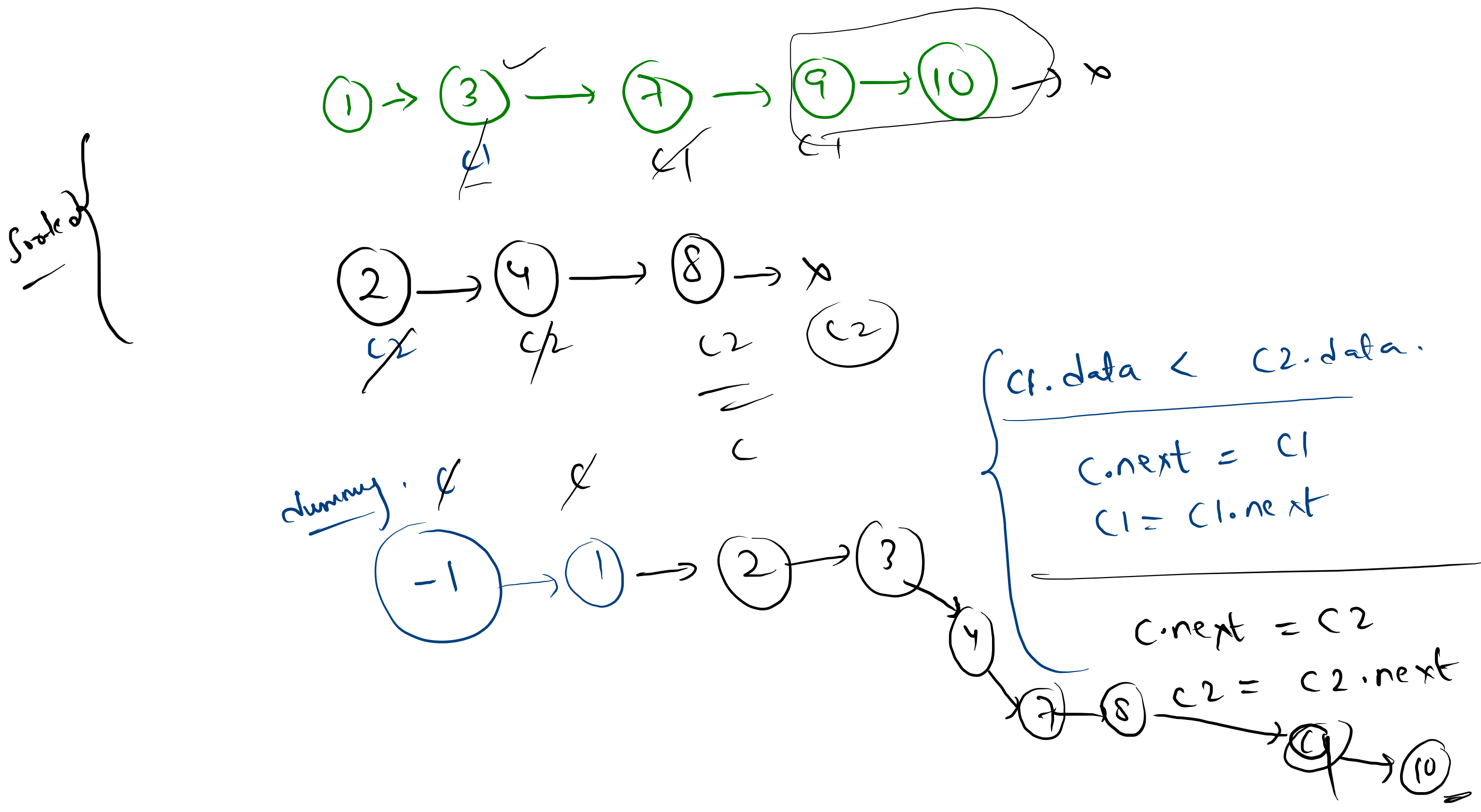
nH

1 → 2 → 3 → 4 → 3 → 2 → 1

nH = m.next

1 → 2 → 3 → 4 → x | 1 → 2 → 3 → x

Merge two sorted linked lists



```
c1.data < c2.data
    c.next = c1
    c1 = c1.next
    ---
    c.next = c2
    c2 = c2.next
```