

Today's Content:

- Basic linked list ✓
 - Insert in a sorted linked list ✓
 - Delete k in a sorted linked list TODO
 - Reverse linked list ✓
 - a) Reverse the first k nodes -
 - b) Reverse all k groups ✓
 - Find the middle of linked list ✓
-

Class Node { // In your language of choice }

int data // variable

Node next // object reference → hold address of Node Object

Node(int n) { // Constructor → used to initialize data members of class

data = n

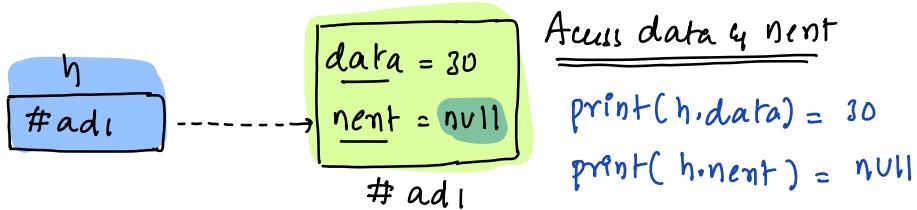
next = NULL

3

3

Node h = new Node(30);

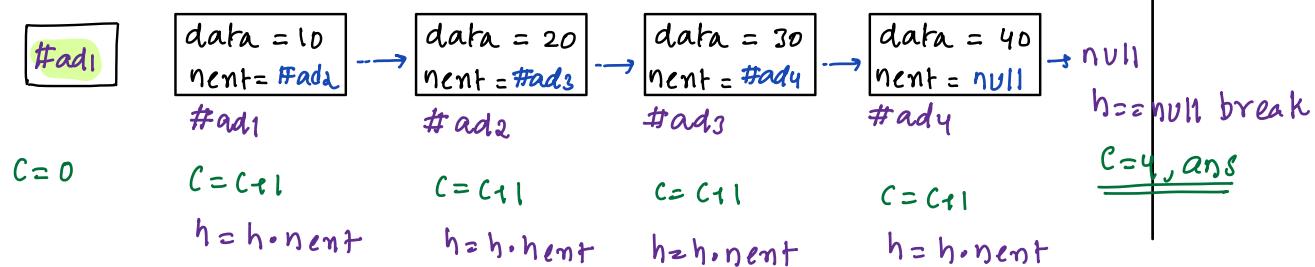
↳ // an object is created & h is an object reference here



Note: h is just holding reference/address of object

print(h) : #ad1

Linked list:



int size(Node h) { TC: O(N) SC: O(1) }

c=0

while(c < h != null) {

 c=c+1

 h=h.next

}

return c

Note: Heavy Interview:

if(h != null) { null.next

 |
 h.data / h.next

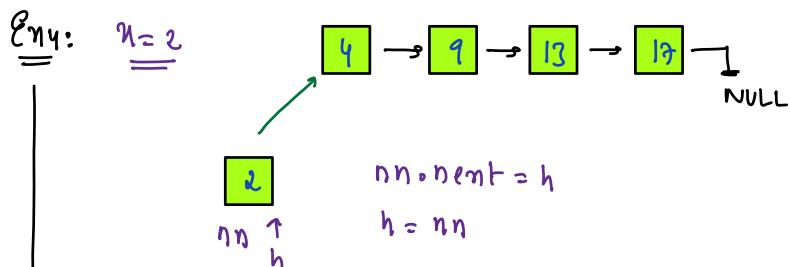
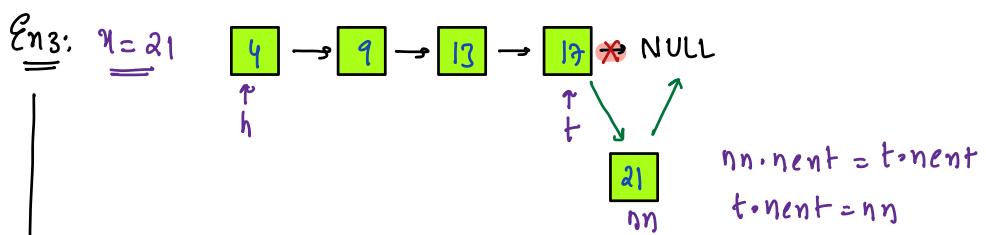
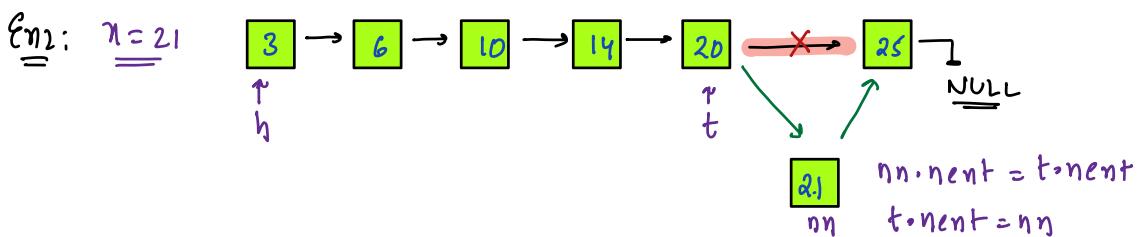
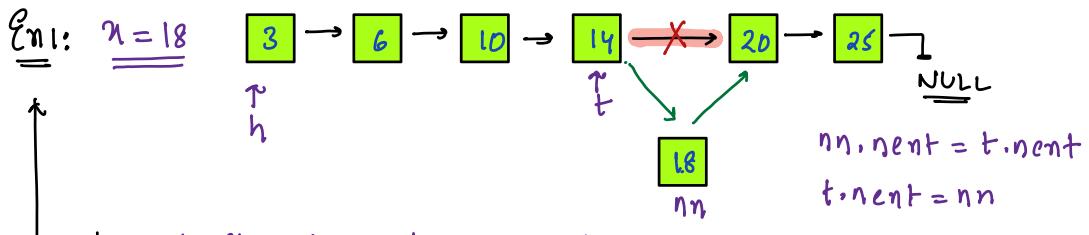
 |

 if(h != null && h.next != null) {

 |
 h.next.data / h.next.next

Problems:

1Q) Given head of a sorted linked, create & insert new node with data=n



Since we might change head return head node of linked list

Node Insert sorted (Node h, int n) { TC: O(N) SC: O(1)}

```
Node nn = new Node(n)
if (h == null) { h = nn, return h}
if (n < h.data) {
    nn.next = h
    h = nn
    return h
}
```

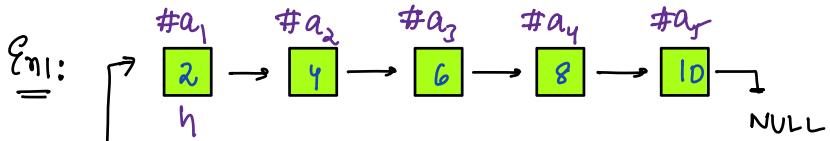
Edge Case: h = null n > 20



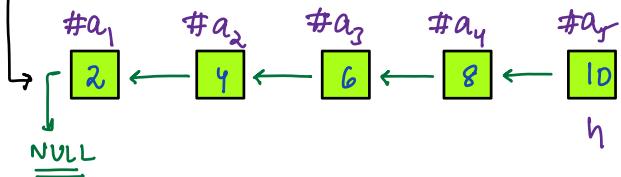
```
Node t = h;
while (t.next != null && t.next.data < n) {
    t = t.next
}
// Insert new node nn after t
nn.next = t.next
t.next = nn
return h
```

Q8) Given a head of linkedlist, reverse linkedlist & return head

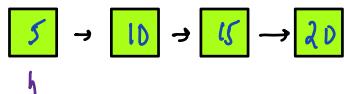
Note: No Extra Space & we cannot change val of node TC: O(N) SC: O(1)



After reversing linked list

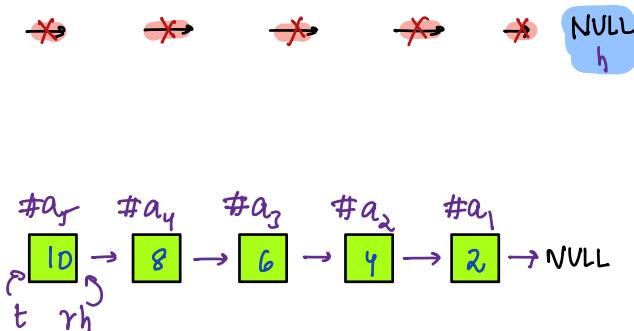


Idea: We inserted 20, 15, 10, 5 at start by 1



// obs: If we insert nodes from start, we will get in reverse order

Trace:



Node rever(Node h) h TC: O(N)

SC: O(1)

Node rh = null

Node t = h

while(h != null) {

 t = h;

 h = h.next

 t.next = null

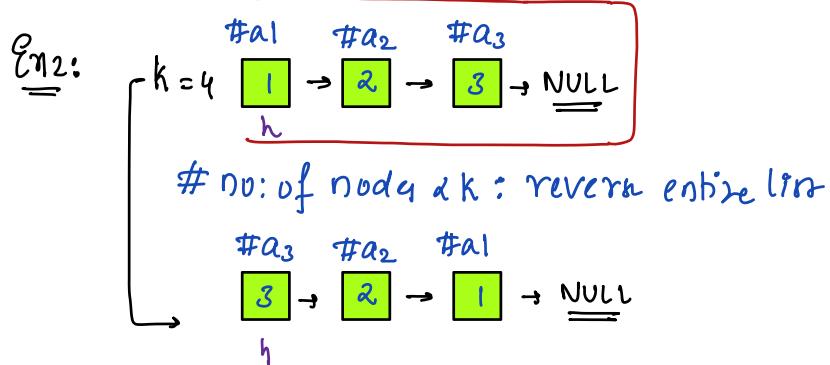
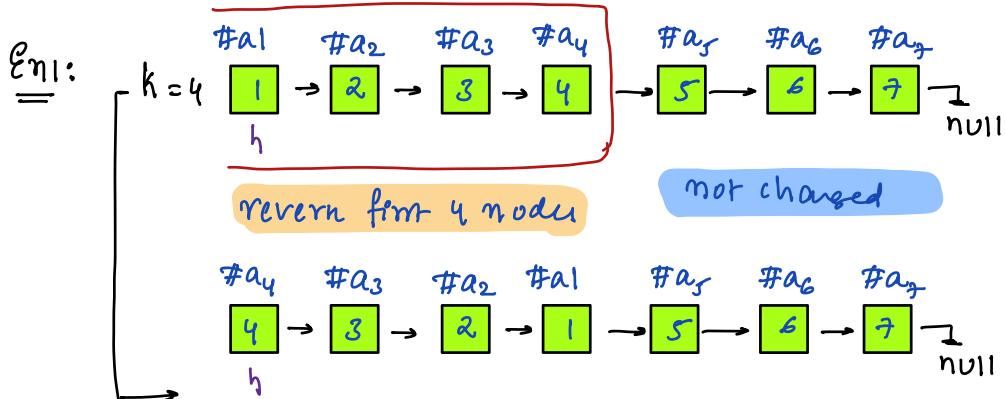
 t.next = rh;

 rh = t

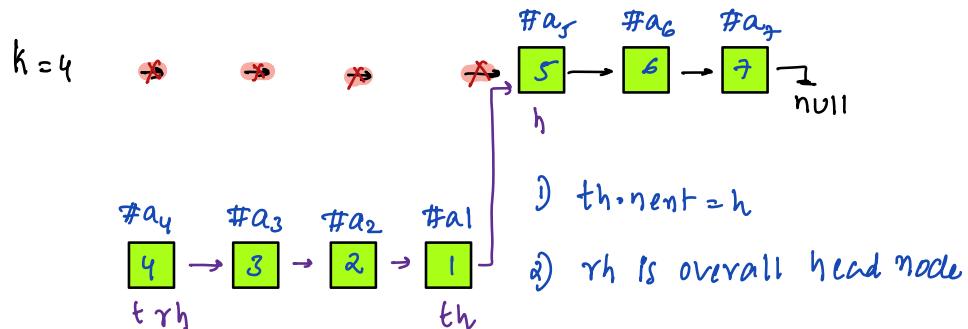
return rh; // head node
of reverse list

Q8) Given head of linked list, reverse first k nodes, given $k > 0$

Note: #no: of nodes $\neq k$, still reverse complete linked list



Trace: Node $th = h$ // Before we reverse, storing head in a temp reference



```
Node rever(Node h, int k){
```

```
if (h == NULL || k <= 1) { return h }
```

Node $th = h$

Node $r.h = \text{null}$, Node $t = h$;

Int c = k

```
while (c > 0 && h != null) {
```

$t = h$
 $h = h \cdot \text{next}$
 $t \cdot \text{next} = \text{null}$
 $t \cdot \text{next} = rh$
 $rh = t$
 $c = c - 1;$

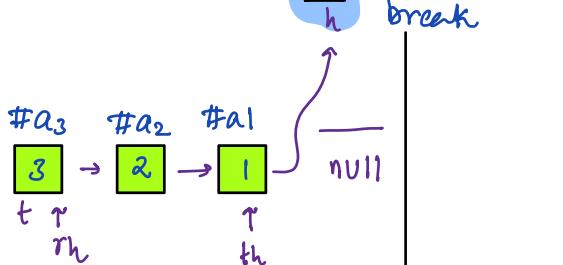
} // Isolating q
 adding at
 Start

the next $= h$] Error can occur, if $h = null$

return rh

Edge Case : To avoid $h != \text{NULL}$

$k=4$ ~~7~~ ~~6~~ ~~5~~ ~~4~~ NULL Since $b = \text{null}$
 b break

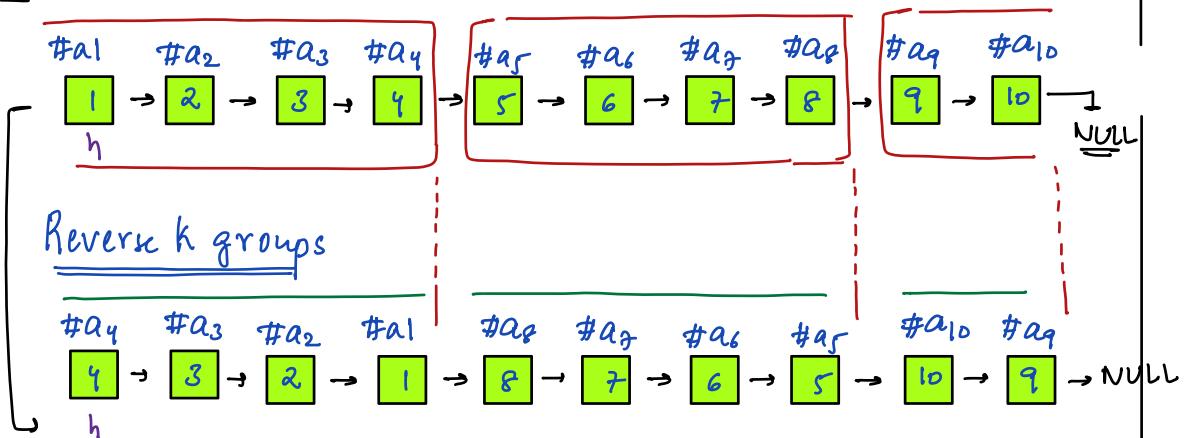


Q8) Given head of linked list, reverse all groups of size: k , $k \geq 0$

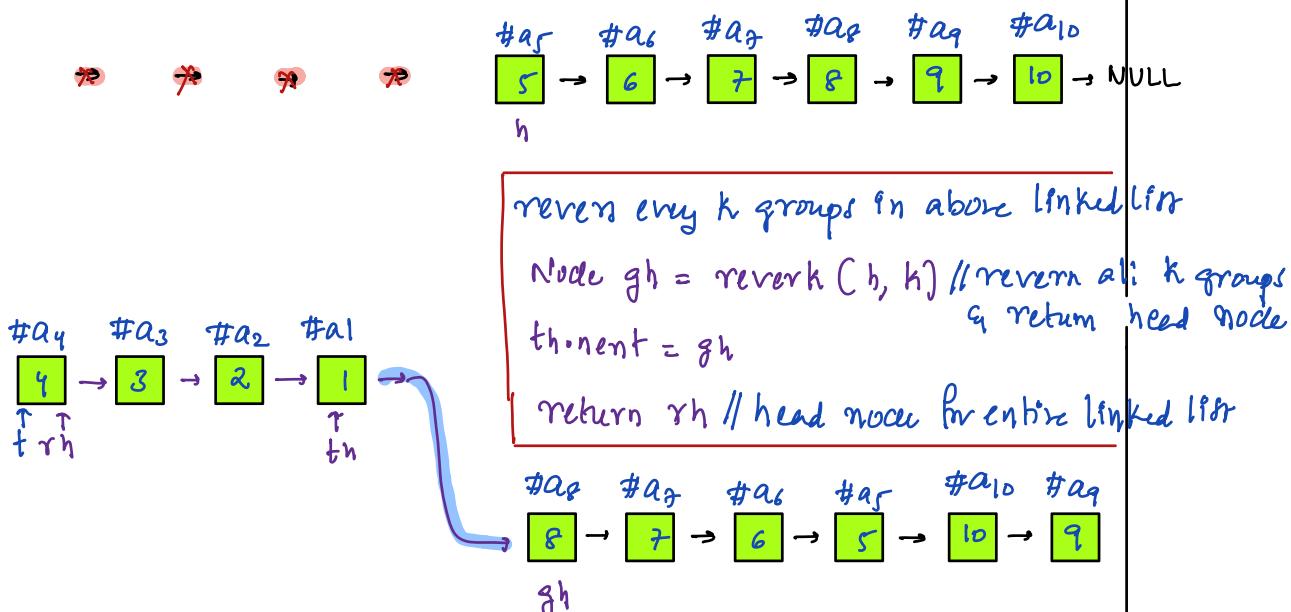
Note: If a group has less than k elements we still need reverse

Note: Recursion based extra space allowed

Ex: $k=4$:



Trace it: $k=4$



Ass: Given a linked list, reverse all groups of size k & return head node

Node reverk (Node h, int \overline{k}) { TC: OCN SC: OCN/k }

if ($h == \text{NULL}$ || $k == 1$) { return h }

Node th = h

Node rh = null, Node t = h;

int c = k

while (c > 0 && h != null) {

 t = h

 h = h.next

 t.next = null

 t.next = rh

 rh = t

 c = c - 1;

} // reversing
from k nodes

TODO: Only iterative code TODO

Node gh = reverk (h, k) } Reversing k groups in remaining

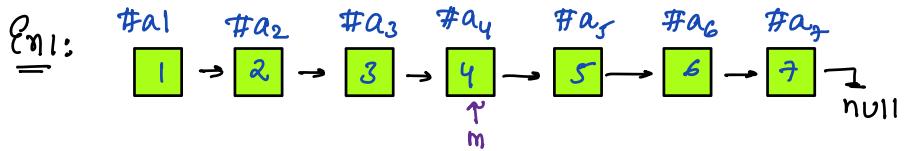
th.next = gh linked list

return rh

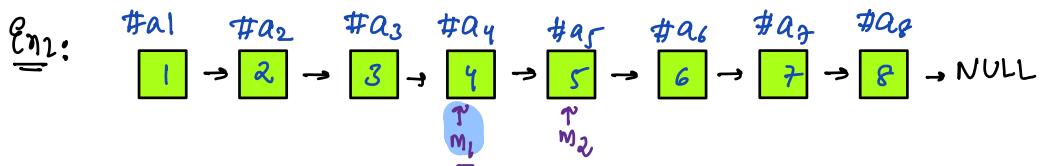
Note: reverse (h, k=2)

Swapping every 2 adj nodes

SQ2) Given head node find the mid of linked list



↑ return 1st mid



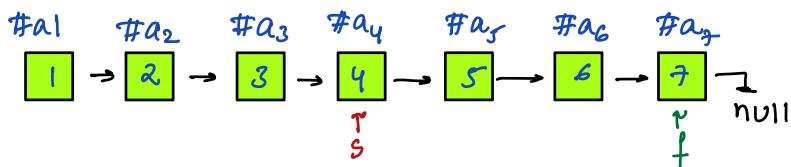
Idea: → Iterate & get size of linked list : n } TC: $O(n)$
→ Again iterate from start & goto center } SC: $O(1)$
→ 2 loops

Idea2: Find mid in a single loop:

Take s, f initialize them at head

: $f = f.next.next$ // fast takes 2 steps

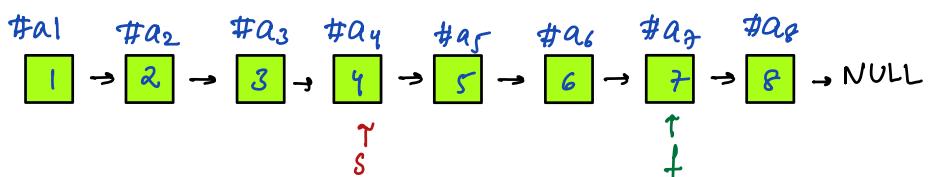
: $s = s.next$ // slow takes 1 step



Obs1: When $f.next == null$, stop & return s

Obs2: When $f.next.next == null$, stop & return s

↳ Use both above conditions



Node mid (Node h) { TC: O(N) SC: O(1)

 if (h == null) { return h; }

 Node s = h;

 Node f = h;

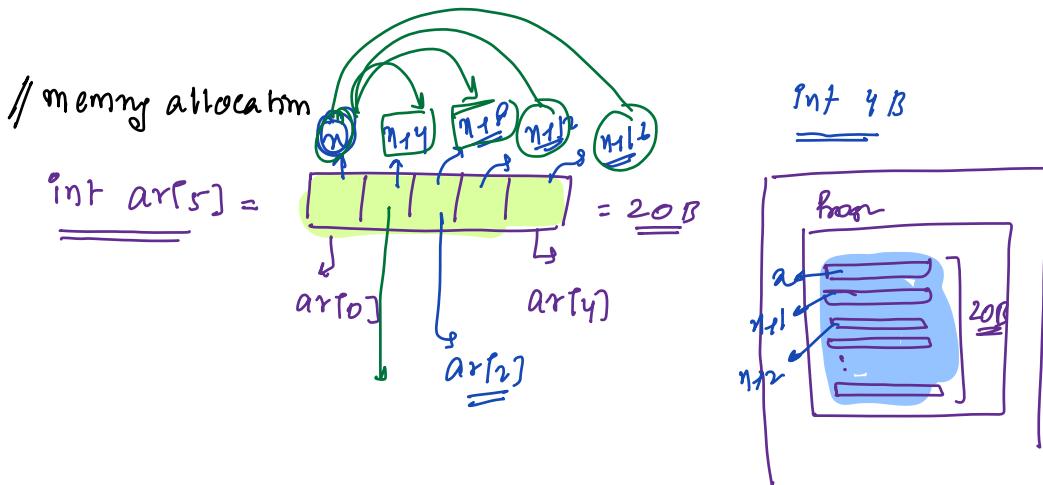
 while (f.next != null && f.next.next != null) {

 s = s.next;

 f = f.next.next;

 }

 return s;



Linked List:

