

## Todays Content

1. Reverse linked list
2. Middle of Linked list
3. Check if Linked list palindrome or not

## Q1 Reverse Linked List

Given a Linked List, reverse entire linked list & return head node.

We can only change next value of a node

Note: We cannot create new node and we cannot change data of node.

SC: O(1)

class Node{

    int data;      # Cannot change value for a node.

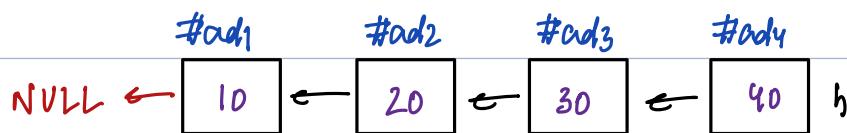
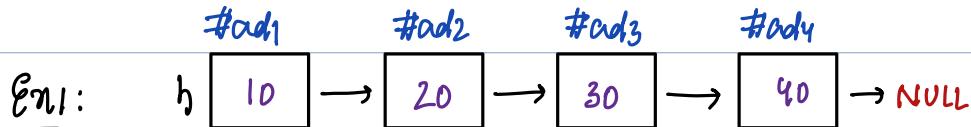
    Node \*next;     # Can change this value for a node

    Node(int n){

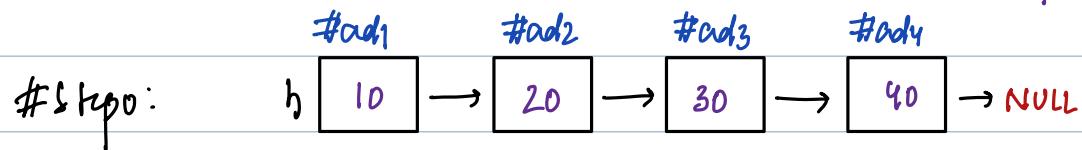
        data = n;

        next = nullptr;

}

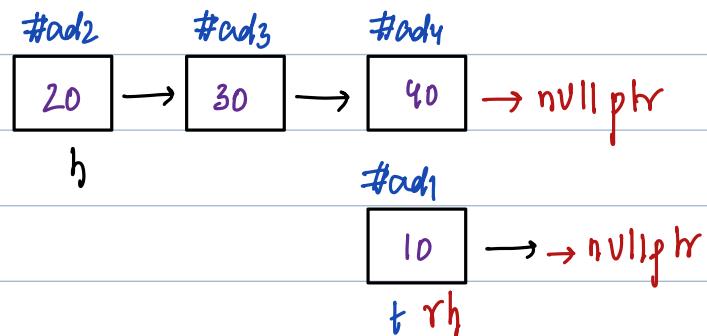


Idea: #Isolate head node & add it at start of reverse.

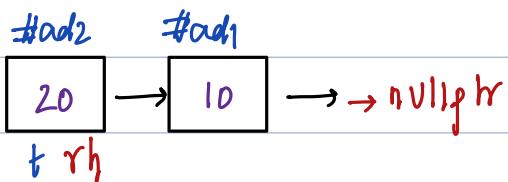
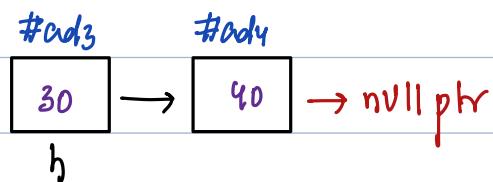


$$r_h = \text{nullptr}$$

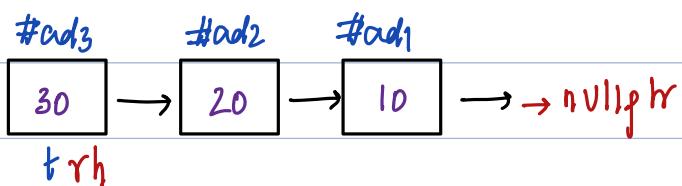
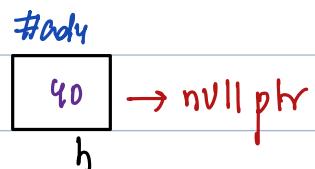
#Step 1:



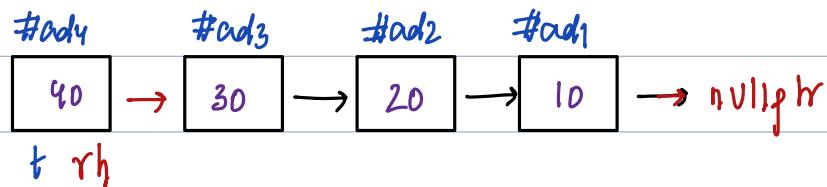
#Step 2:



#Step 3:



#Step 4:



Node\* reverse(Node\*h) { TC: O(N) SC: O(1) }

Node \*rh = nullptr;

while(h != nullptr) {

Node \*t = h;

h = h->next;

t->next = rh;

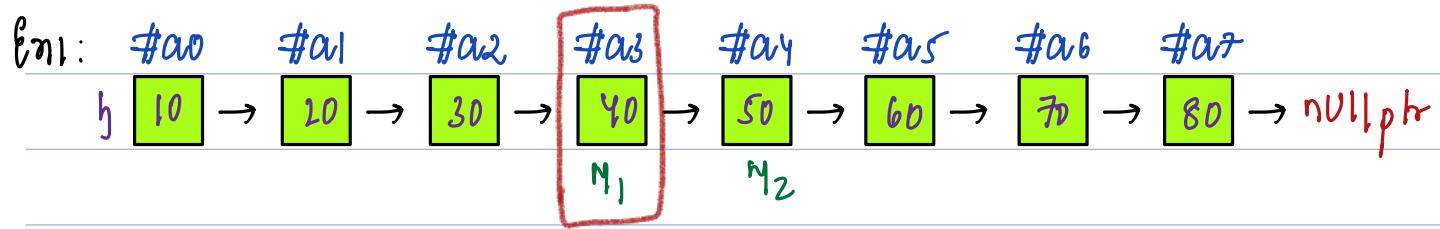
rh = t;

3

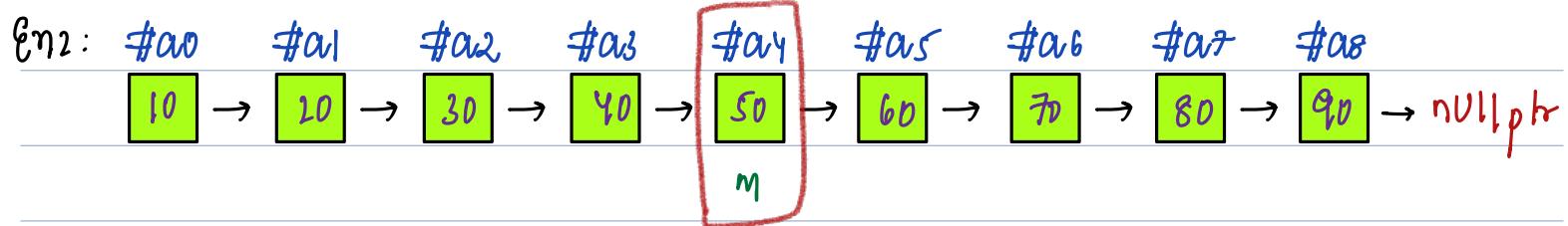
return rh;

3

10 Given head node find mid node of linked list



# of Even length return 1<sup>st</sup> mid.



Ideas:

1. Calculate length of linked list = N → 1<sup>st</sup> loop

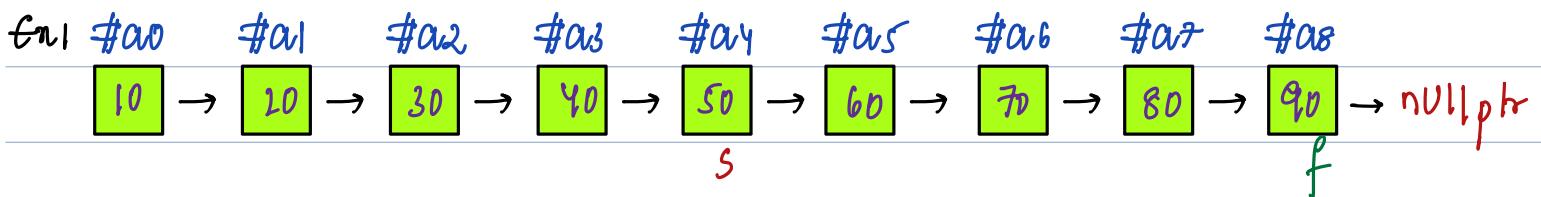
2. Get index  $(N-1)/2$  & return node address → 2<sup>nd</sup> loop.

Tc:  $O(N + N/2) \Rightarrow O(N)$  SC:  $O(1)$

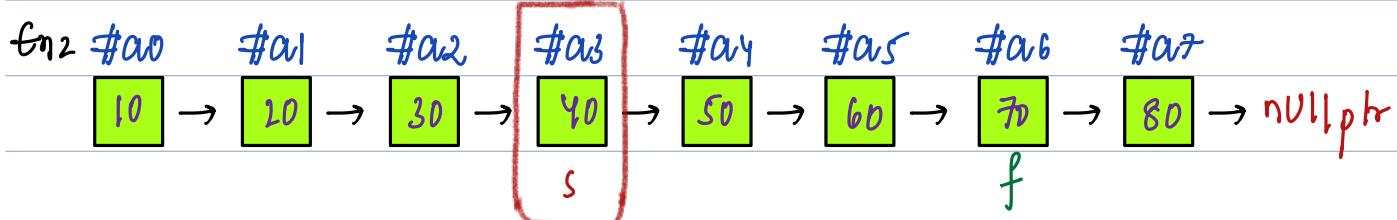
Ideas:

1. Take a  $s$  and  $f$  reference initialized at head.
2. At each iteration  $s = s \rightarrow next$  &  $f = f \rightarrow next \rightarrow next$
3. By the time fast reaches end slow reaches mid.

Dry Run:

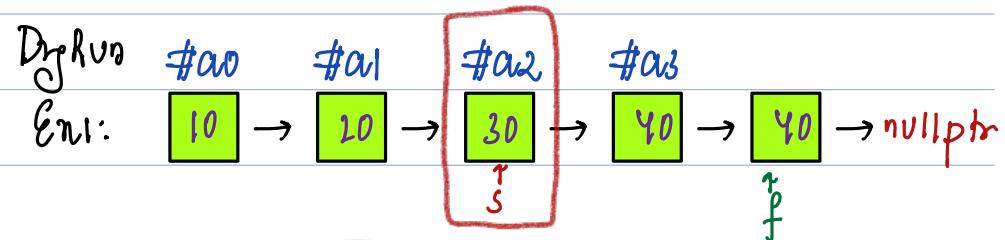


if ( $f \rightarrow next == \text{nullptr}$ ) { if (#s is at mid)



if ( $f \rightarrow next \rightarrow next == \text{nullptr}$ ) { if (#s is at 1^"mid)

Tl:  $O(N)$  Sc:  $O(1)$



Node\* Mid(Node\* h) {

    Node \*s = h, \*f = h;

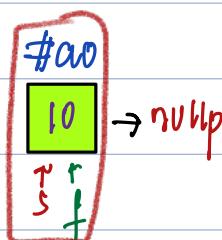
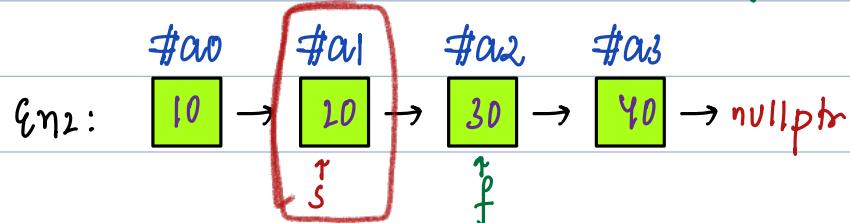
    while (f != nullptr) {

        if ( $f \rightarrow next == \text{nullptr} \text{ || } f \rightarrow next \rightarrow next == \text{nullptr}$ ) { return s; }

        s = s  $\rightarrow$  next;

        f = f  $\rightarrow$  next  $\rightarrow$  next;

    return f;



38: Given a linked list, check if it's palindrome or Not?

#Note: No Extra Space

Ex: #a0    #a1    #a2    #a3    #a4    #a5    #a6    #a7    #a8

| 10 → 20 → 30 → 40 → 50 → 40 → 30 → 20 → 10 → null, ptr

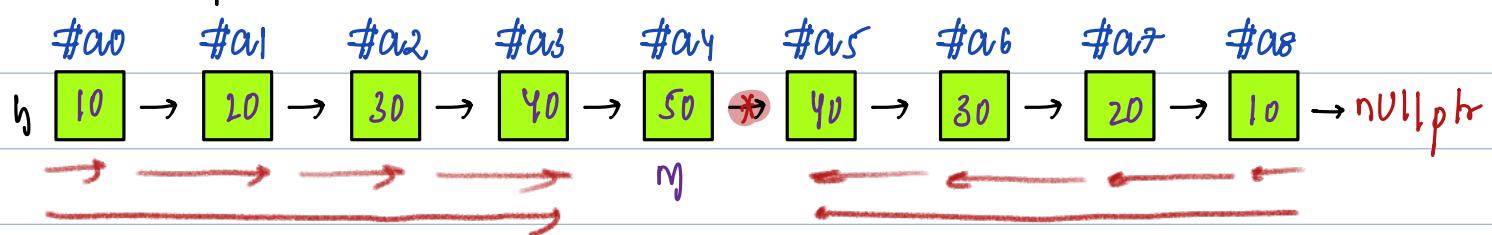
Idea:

#a0    #a1    #a2    #a3    #a4    #a5    #a6    #a7    #a8

| 10 → 20 → 30 → 40 → 50 → 40 → 30 → 20 → 10 → null, ptr



#Idea Step 1: Calculate mid.

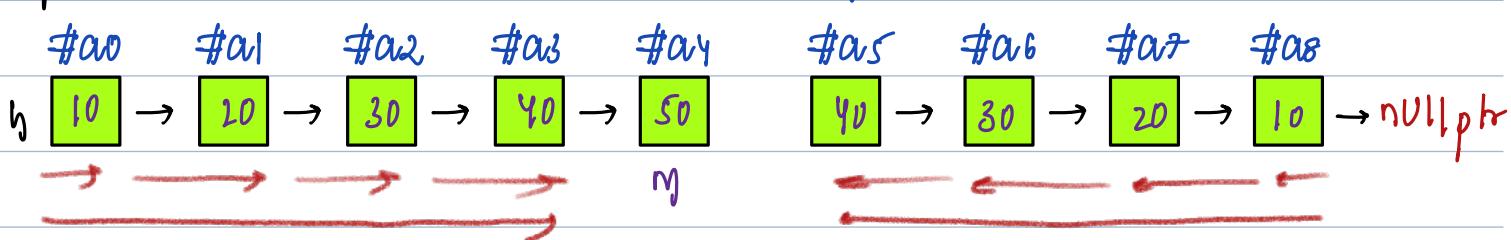


Node \* $m = \text{mid}(h)$

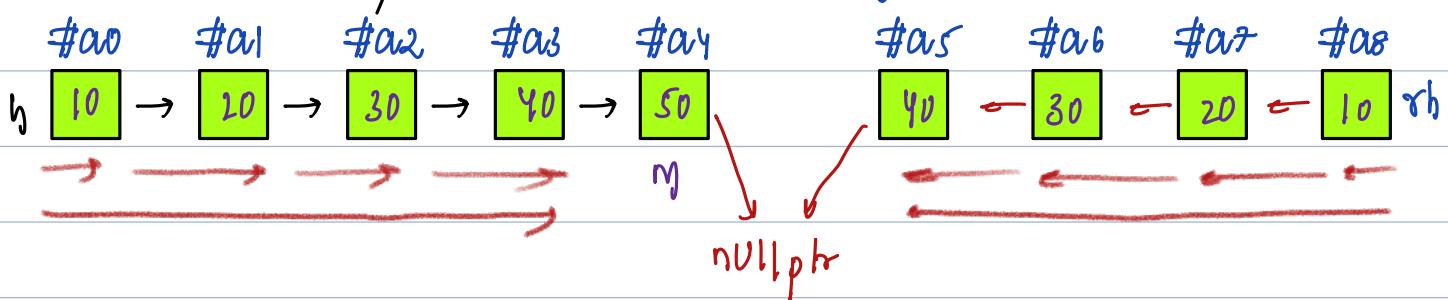
Node \* $r_h = m \rightarrow \text{next}$

$M \rightarrow \text{next} = \text{NULLptr}$

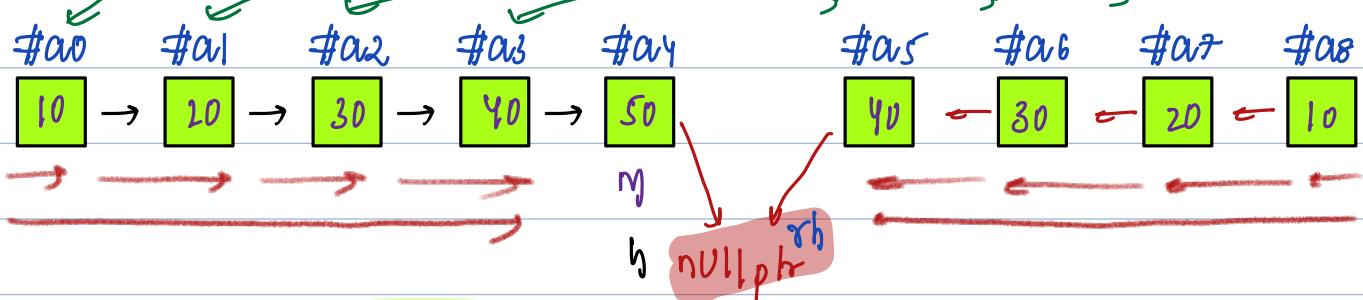
#Step 2:



$r_h = \text{reverse}(rh)$ ; #if will return head of reverse, & for in rh



#Step 3:



while ( $h \neq \text{NULLptr}$  &&  $r_h \neq \text{NULLptr}$ ) {

    if ( $h \rightarrow \text{data} \neq r_h \rightarrow \text{data}$ ) { return false }

$h = h \rightarrow \text{next}$

$r_h = r_h \rightarrow \text{next}$

return true;

boolean ispal(Node \*h) { TC:  $O(N+N+N) = O(N)$  SC:  $O(1)$

Node \*m = mid(h)

Node \*rh = m->next;

m->next = nullptr;

rh = reverse(rh);

while(h != nullptr && rh != nullptr) {

if(h->data != rh->data) { return false; }

h = h->next;

rh = rh->next;

return true;

3