

Today's 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;
```

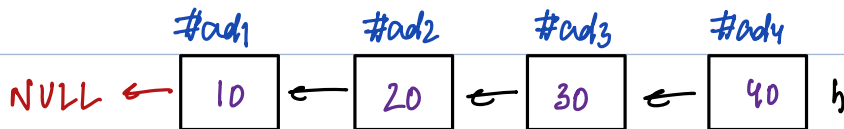
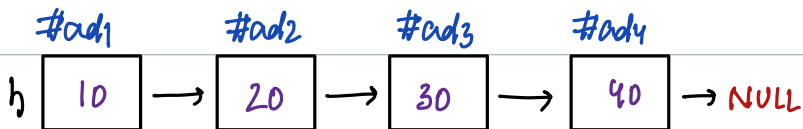
Can change this value for a node

```
    Node(int n) {
```

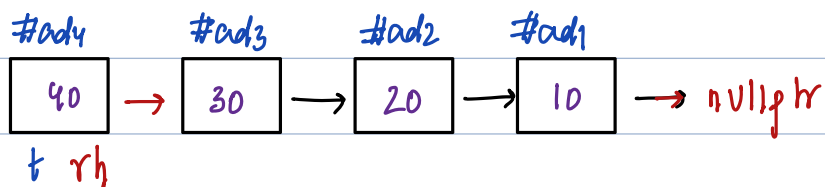
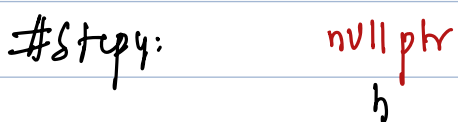
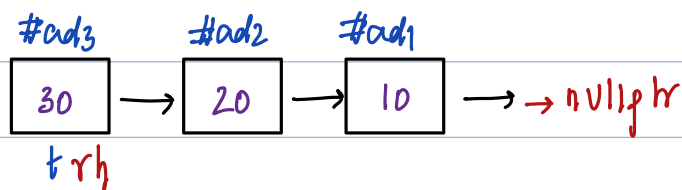
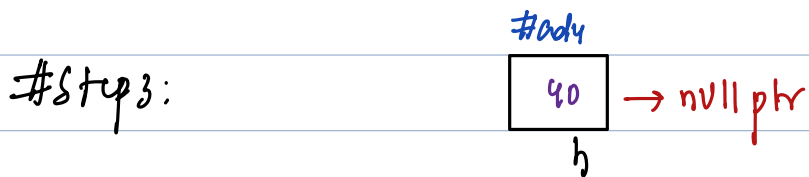
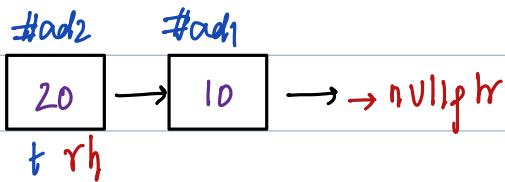
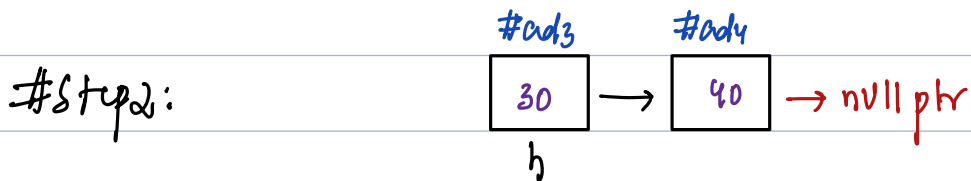
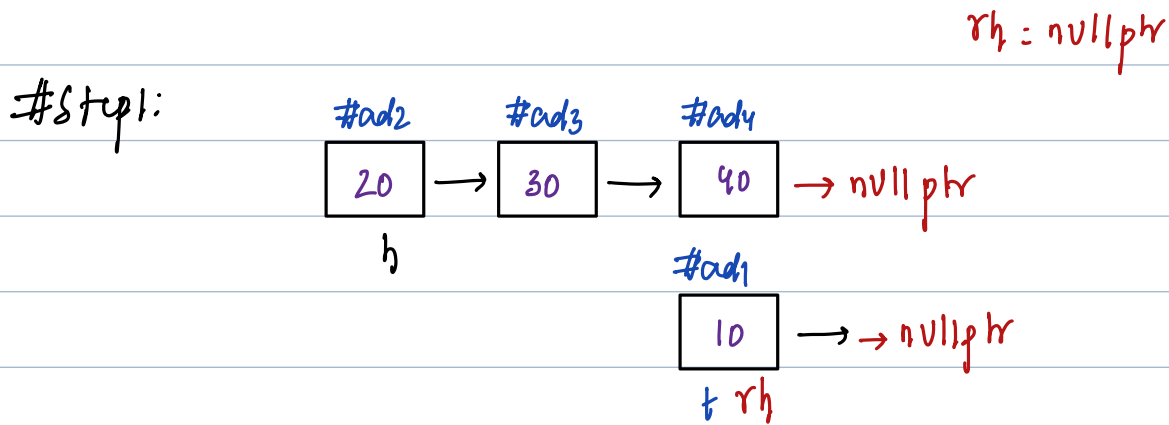
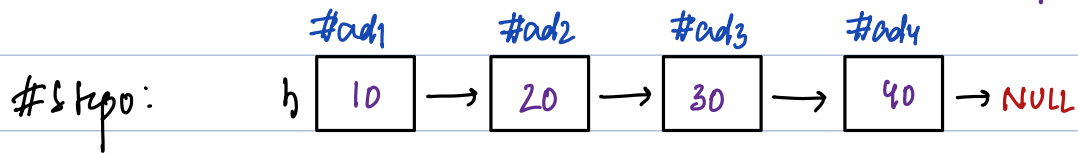
```
        data = n;
```

```
        next = nullptr;
```

Ex1:



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



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;

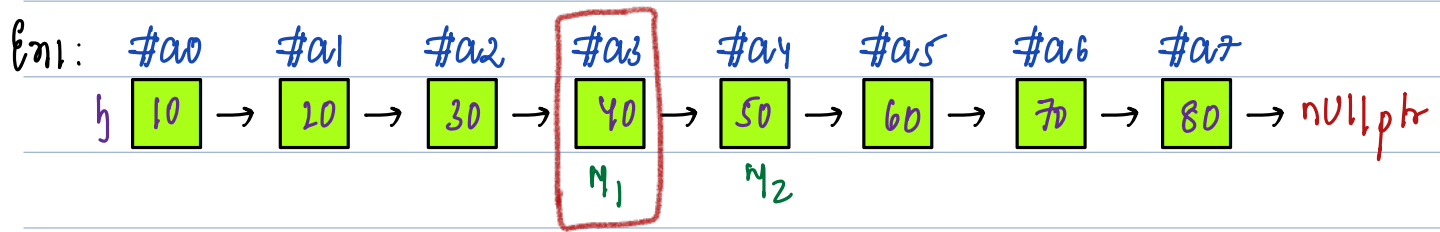
rh = t;

}

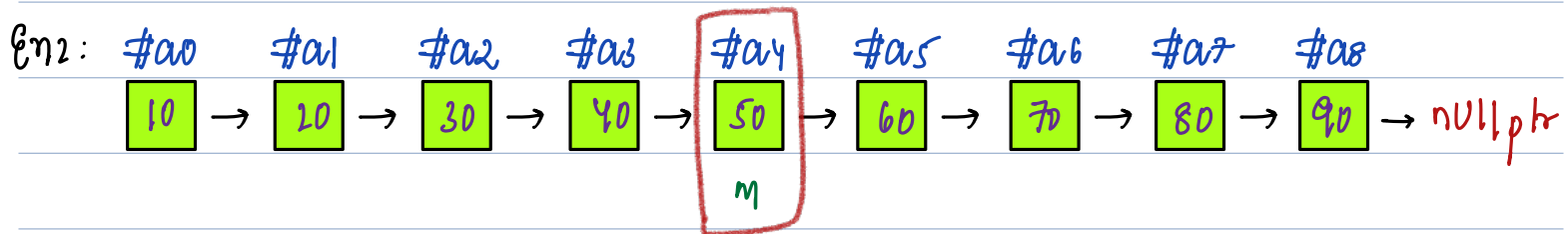
return rh;

}

1Q Given head node find mid node of linked list



If Even length return 1st mid.



Idea:

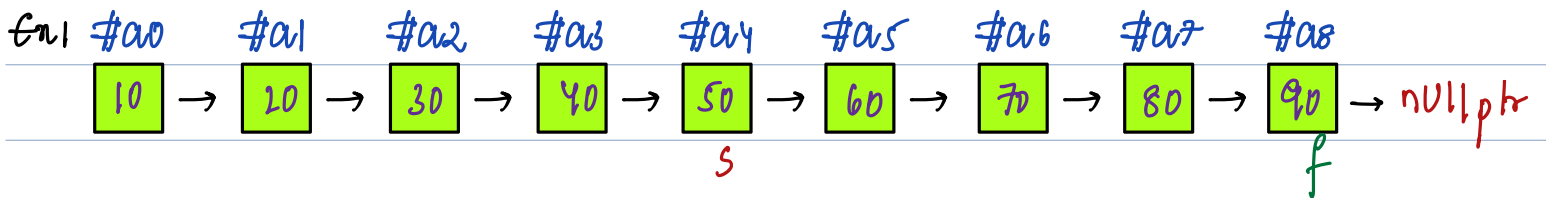
1. Calculate length of linked list = N → 1st loop
2. Get Index $(N-1)/2$ & return node address → 2nd loop.

T.C: $O(N + N/2) \approx O(N)$ S.C: $O(1)$

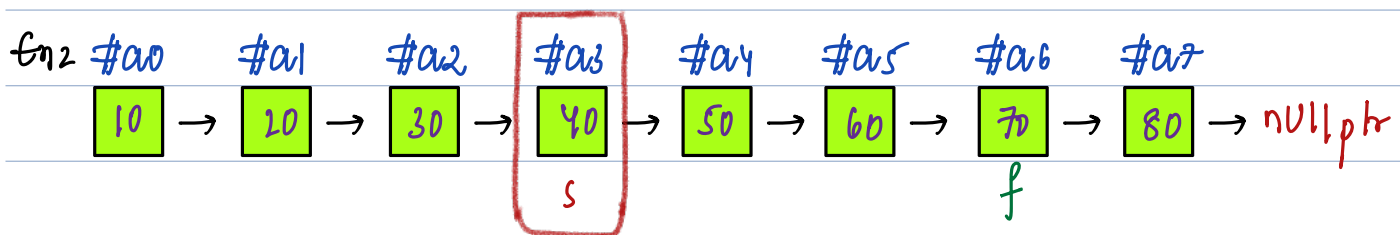
Idea:

1. Take a s and f reference initialize at head.
2. At each iteration $s = s \rightarrow \text{next}$ & $f = f \rightarrow \text{next} \rightarrow \text{next}$
3. By the time f reaches end s reaches mid.

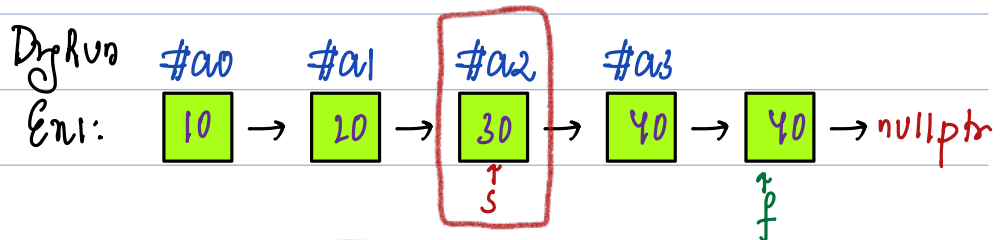
Dry Run:



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



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



T.C: $O(N)$ S.C: $O(1)$

Node* mid(Node* h) {

Node* $s = h$, $f = h$

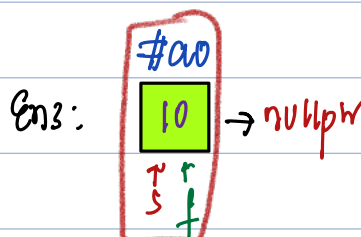
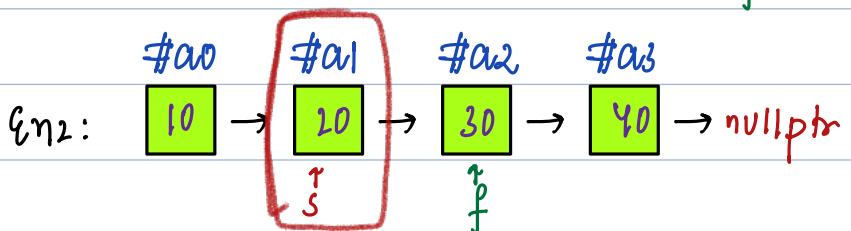
while ($f != \text{null}$) {

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

$s = s \rightarrow \text{next}$;

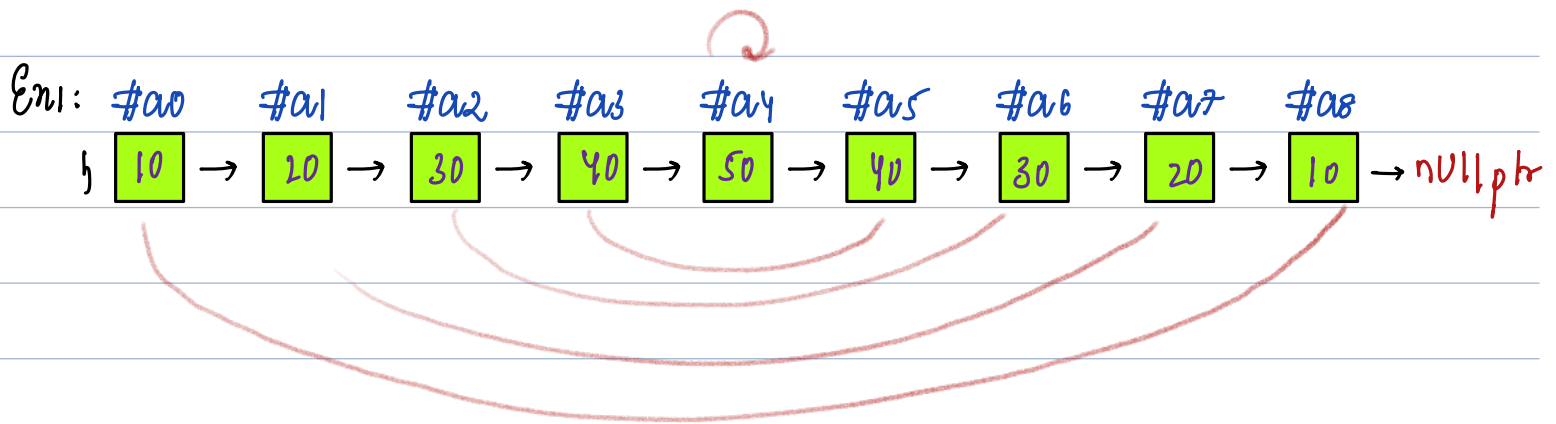
$f = f \rightarrow \text{next} \rightarrow \text{next}$;

return f # of $f == \text{null}$

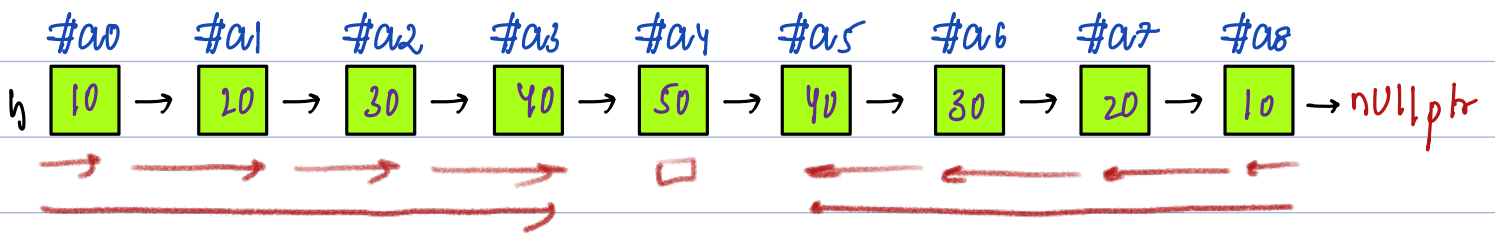


38: Given a linked list, check if it's palindrome or not?

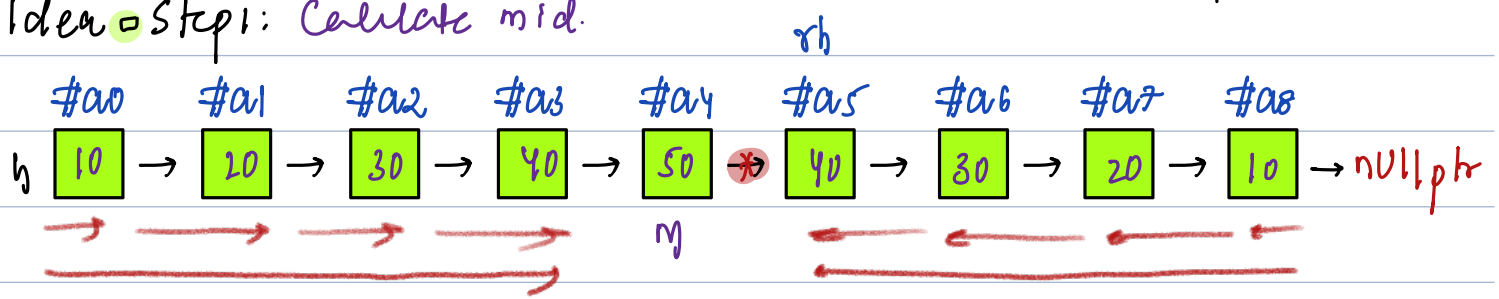
#Note: No Extra Space



Idea:



Idea = Step 1: Calculate mid.

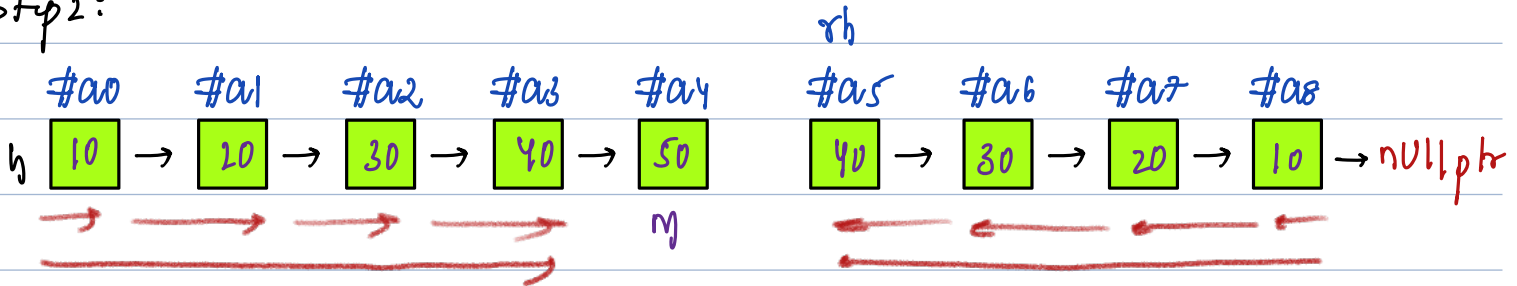


Node $*m = \text{mid}(h)$

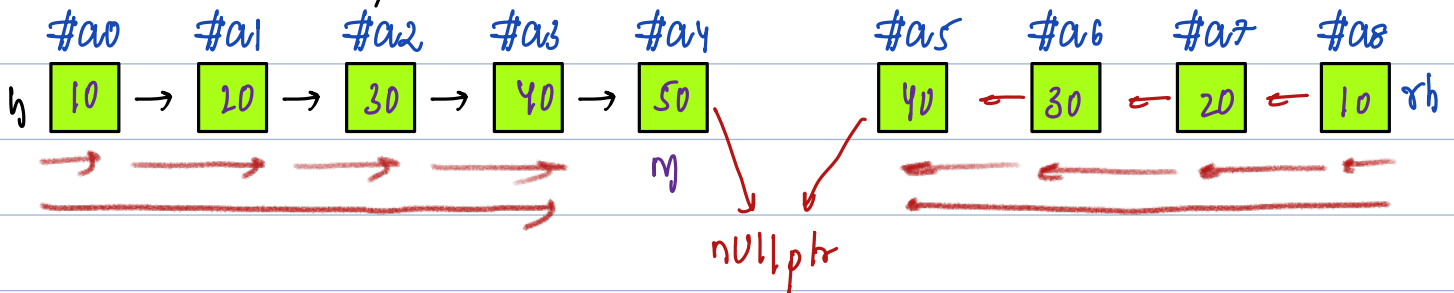
Node $*rh = m \rightarrow \text{next};$

$m \rightarrow \text{next} = \text{nullptr};$

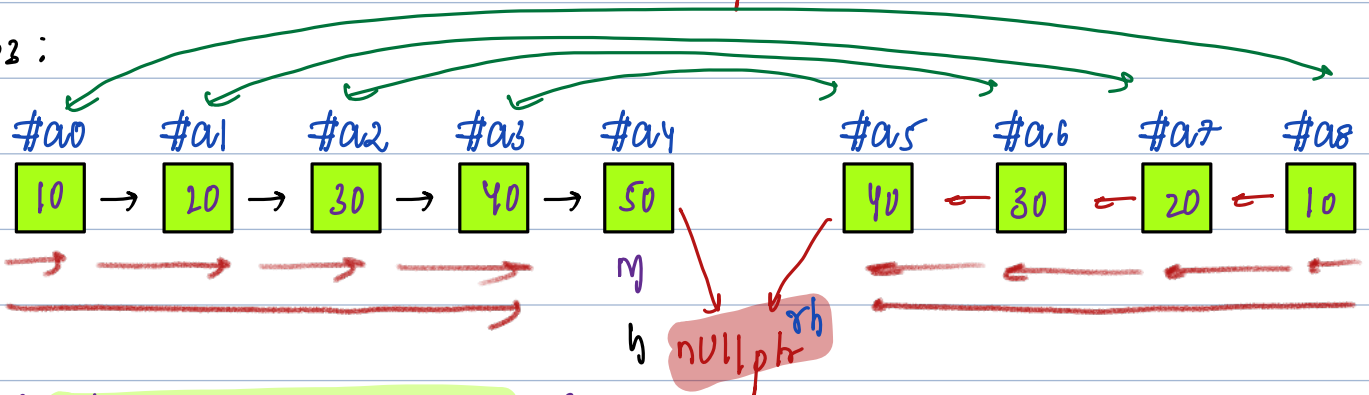
Step 2:



$rh = \text{reverse}(rh);$ # it will return head of reverse, store in rh



Step 3:



while($h \neq \text{nullptr}$ & $rh \neq \text{nullptr}$) {

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

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

$rh = rh \rightarrow \text{next};$

return true;

boolean ispal(Node *h) { T.C: $O(N+N+N) = O(N)$ S.C: $O(1)$

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;