Reverse a LL

Check whether LL has palindrome

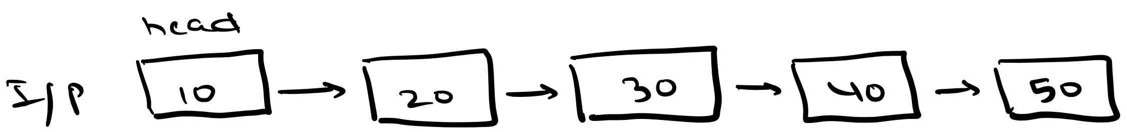
Find middle element of LL

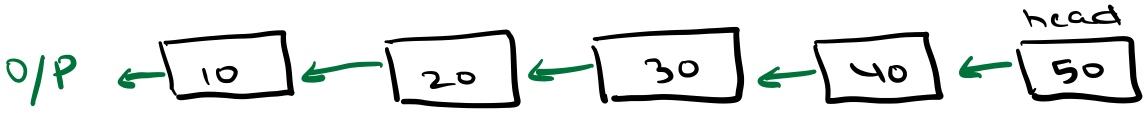
Merge 2 sorted LL

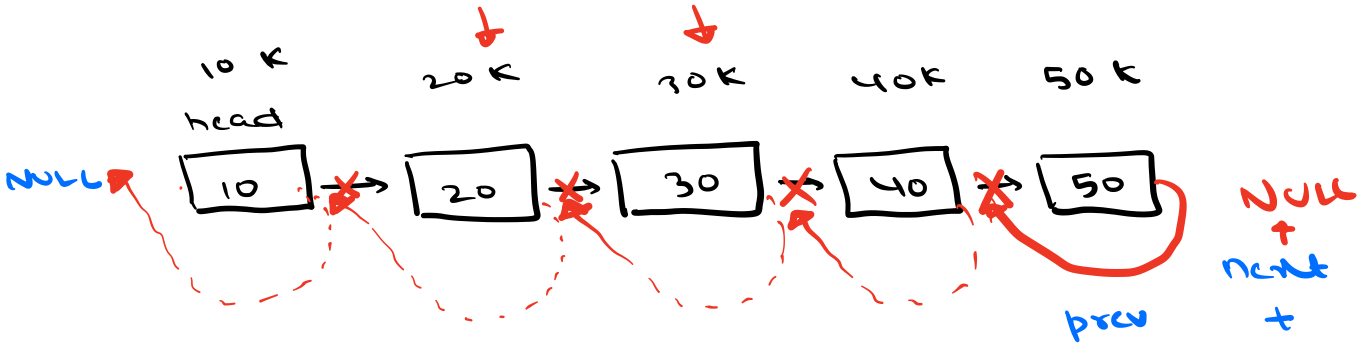
Sort a LL

Find cycle in LL

1. Reverse a LL





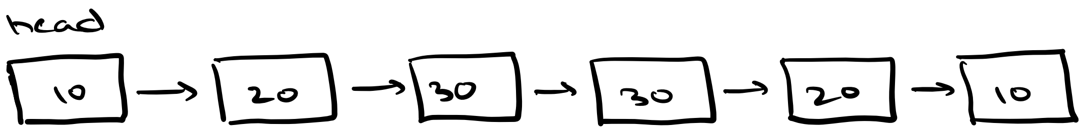


```
Node reverse(Node head) {
        Node temp = head                 Node next=NULL
        Node prev = NULL
    while(temp != NULL){
          next = t.next
        t. next = prev
          prev = t          } jump to
            t = next           next node
    }
    return prev
}
```

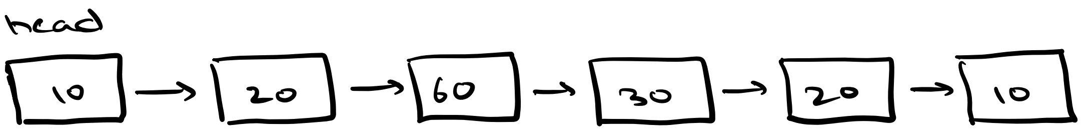
TC: O(N)

SC: O(1)

2. Given a LL, check if its palindrome.



True



False

① list = reverse (list)

② symmetrical around centre

Sol 1 : 1) Create a copy of LL

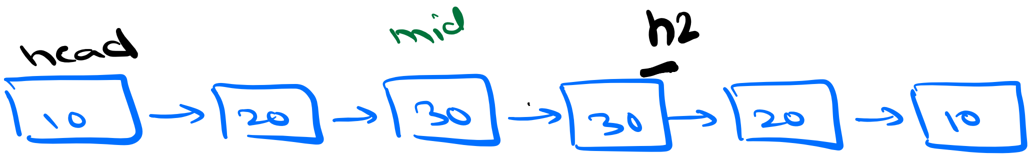
2) Reverse it

3) Compare data of nodes one by node

TC : O(N)

SC : O(N)

Sol 2 : SC → O(1)



6/2 = 3

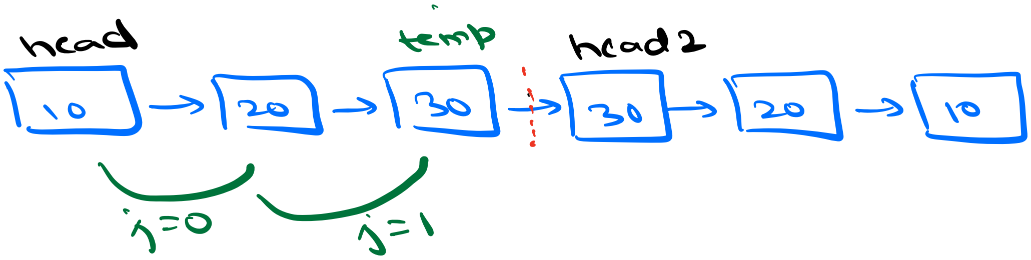
6

Step 1 : size of LL N

```
int cnt = 0
Node temp = head
while (temp ! = NULL)
    cnt++
    temp = temp.next
}
```

Step 2 : Reach middle node

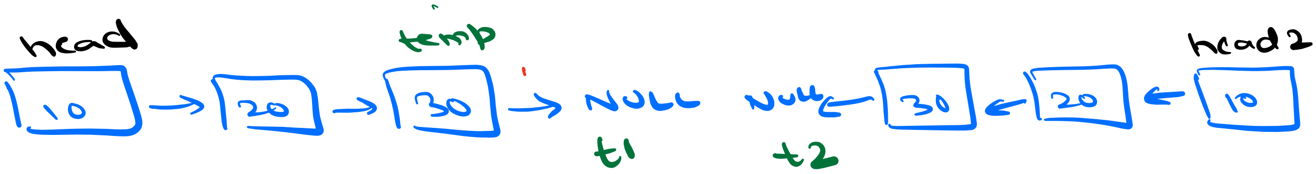
```
int mid = cnt/2

for (jump = 0; jump < mid-1; jump++){
    temp = temp.next                      N/2
}
```



```
Node head2 = temp.next
temp.next = NULL
```

Step 3 : Reverse 2nd half

```
head2 = reverse (head2)                   N/2
```



```
Node t1 = head
Node t2 = head2

while (t1 != NULL && t2 != NULL) {
    if (t1.data != t2.data)
        return false
    t1 = t1.next
    t2 = t2.next
}

return true
```

N/2

TC : O(N)

SC : O(1)

head temp head2

10 → 20 → 30 → 50 → 30 → 20 → 10

n = 7

$n/2 = 7/2 = 3$

head

10 → 20 → 30 → N

t1

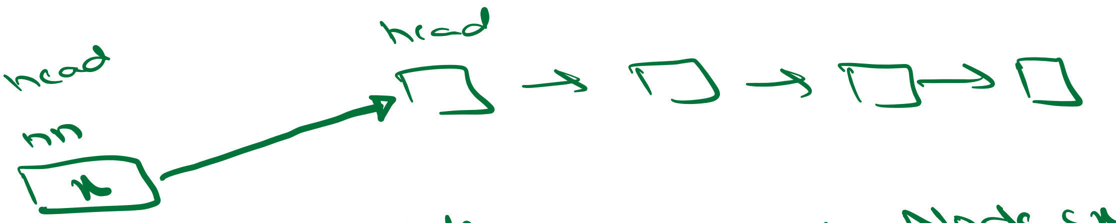
head 2

10 → 20 → 30 → 50

t2

Delete a node from LL → TC : O(N)

Insert a node at head of LL TC : O(1)



Node nn = new Node(x)

nn. next = head

head = nn

---

3. Given LL, find middle element.

Odd

head, mid, cnt=5

$a_1 \to a_2 \to a_3 \to a_4 \to a_5$ Mid= 3

Even

head, mid, cnt=6

$a_1 \to a_2 \to a_3 \to a_4 \to a_5 \to a_6$

mid.next

mid' = 3

Sol 1 a) Get size of LL → cnt

b) mid = $\frac{cnt+1}{2}$

TC : O(N)

SC : O(1)

c) Jump mid-1 times

Sol 2 : Do it in 1 iteration

odd

$$\overset{\text{head}}{a_1} \to a_2 \to a_3 \to \overset{\text{middle}}{\underset{s}{a_4}} \to a_5 \to a_6 \to \underset{f}{a_7}$$

when f.next = null STOP

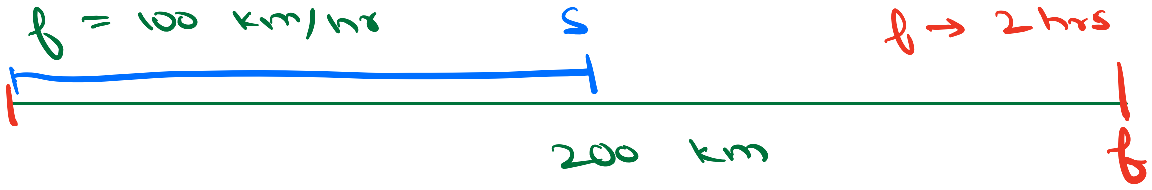
s→1 steps
f→2 steps

Even

$$\overset{\text{head}}{a_1} \to a_2 \to a_3 \to \overset{\text{middle}}{\underset{s}{a_4}} \to a_5 \to a_6 \to \underset{f}{a_7} \to a_8$$

when f.next.next = NULL STOP

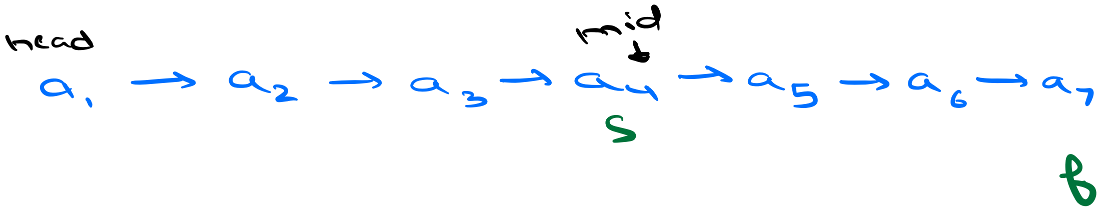
s = 50 km/hr
f = 100 km/hr

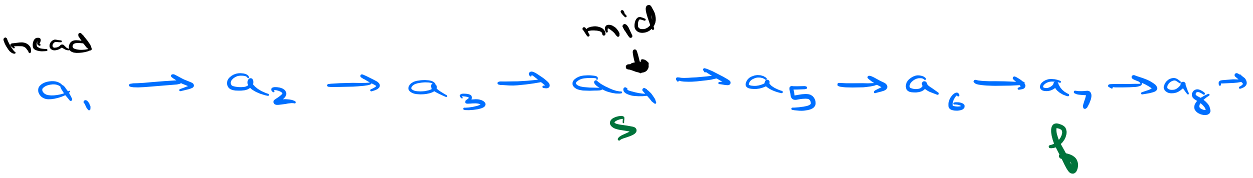


```
Node mid (Node h) {
    if (h == NULL) return NULL
        Node s = h
        Node f = h

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

              s = s.next
              f = f.next .next
        }

        return s
}
```

TC : O(N)

SC : O(1)





4. Given 2 sorted LL, merge them into a single LL. (sorted)

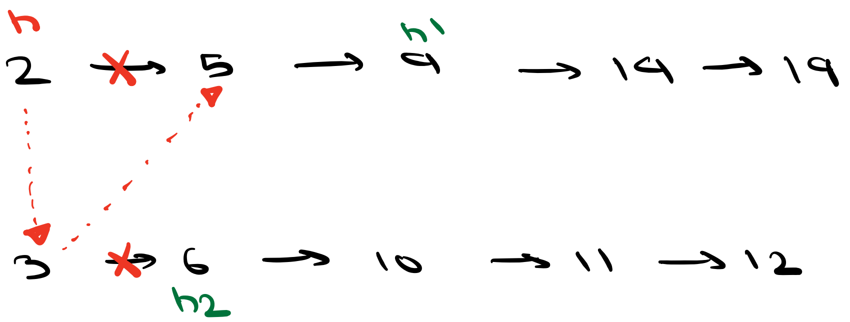
eg. LL1) $\overset{h1}{2}$ → 4 → 6 → 8 → 10

LL2) $\underset{h2}{1}$ → 3 → 5 → 7 → 9

O/P $\overset{h}{1}$ → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10

---

eg

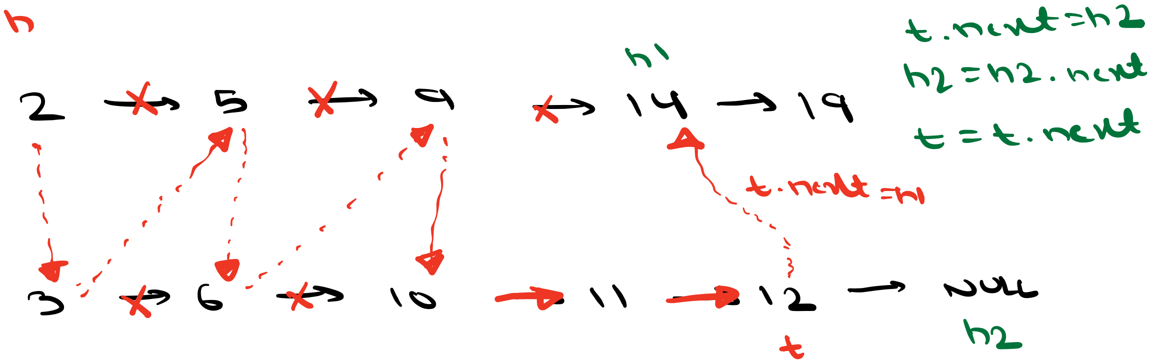


3.next = 5

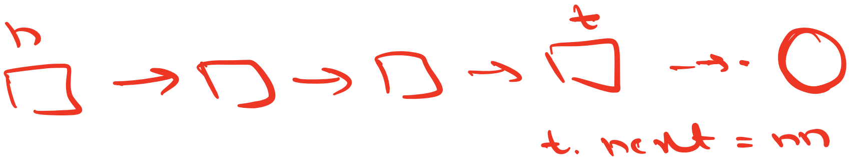
h.next = h2

$\underline{\underset{2}{h}\,.\underset{3}{next}}$ .next = h1

$\overset{h}{2}$ → 3 → 5



t.next=h2
h2=h2.next
t=t.next



t. next = nn

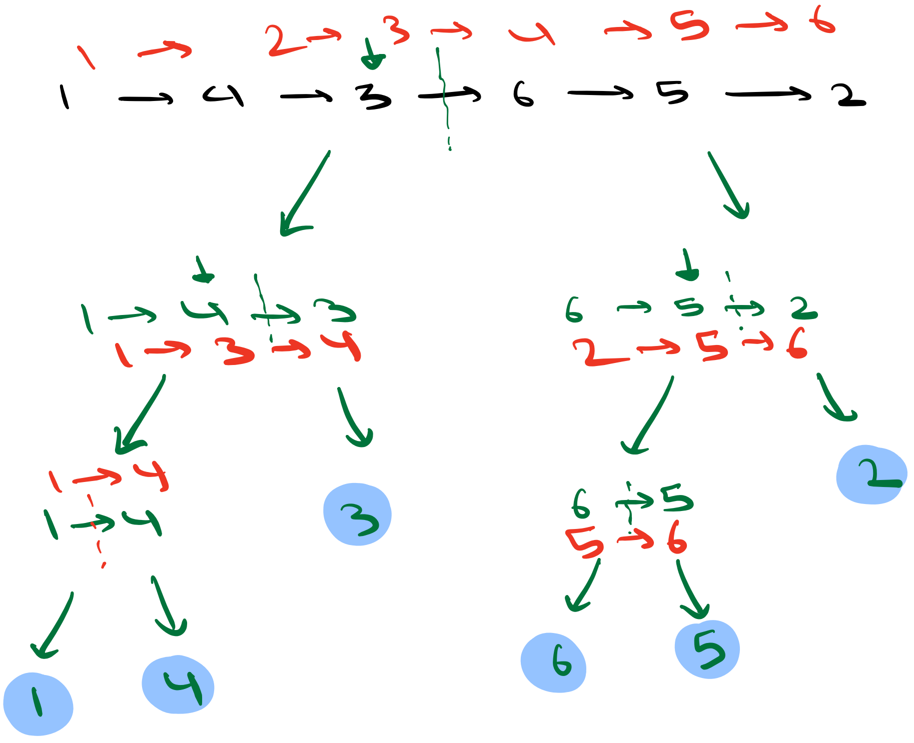
t.next =h1
h1 =h1. next
t = t.next

```
Node merge(Node h1, Node h2) {
    if (h1 == null)  return h2
    if (h2 == null)  return h1
      Node h, t
  if (h1.data <= h2.data) { h = h1   t = h1
                            h1 = h1.next }
  else {
        h = h2      t = h2
        h2 = h2.next
  }
     while (h1 != NULL && h2 != NULL) {
          if (h1.data ≤ h2.data) {
               t.next = h1
               h1 = h1.next
               t = t.next
          }
          else {
               t.next = h2
               h2 = h2.next
               t = t.next
          }
     }
     if (h1 == null)
               t.next = h2
     else if (h2 == null)
               t.next = h1
     return h
}
```

TC: O(M+N)

SC: O(1)

5. Sort a LL



Merge sort on LL

h
I/P 1 → 4 → 3 → 6 → 5 → 2

O/P 1 → 2 → 3 → 4 → 5 → 6
h

h1 m h2

1 → 4 → 3 → 6 → 5 → 2

Step 1: Find middle

h2 = m.next

m.next = NULL

h1 h2

1 → 4 → 3   6 → 5 → 2

Step 2: h1 = mergesort (h1)

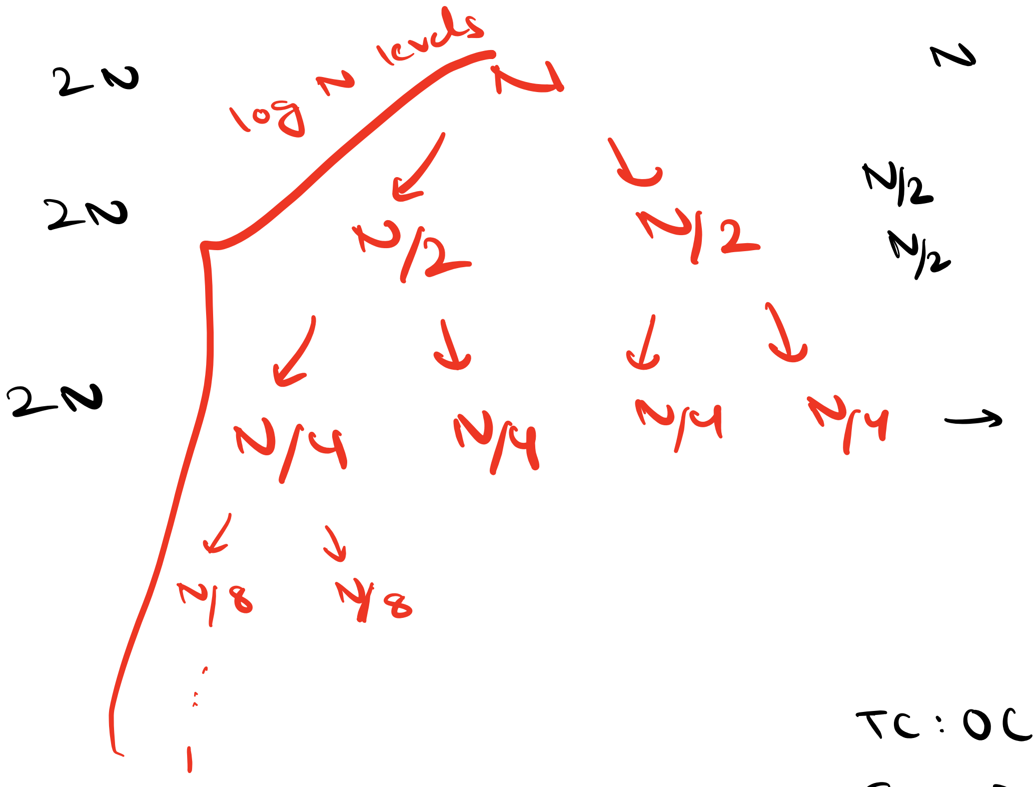
h2 = merge sort (h2)

h1 h2

1 → 3 → 4   2 → 5 → 6

Step 3: Merge both sorted LL

```
Node mergesort (Node h1) {
    if (h1 ==NULL || h1.next ==NULL)
        return h1
    Node m = middle (h1) →
    Node h2 = m. next
      m. next = NULL
        h1 =    mergesort (h1)
        h2 =    mergesort (h2)
         h  =   merge (h1, h2)
        return h
}
```



Middle		merge
N	+	N
N/2		N/2
N/2		N/2

TC : $O(N \log_2 N)$

SC : $O(\log_2 N)$