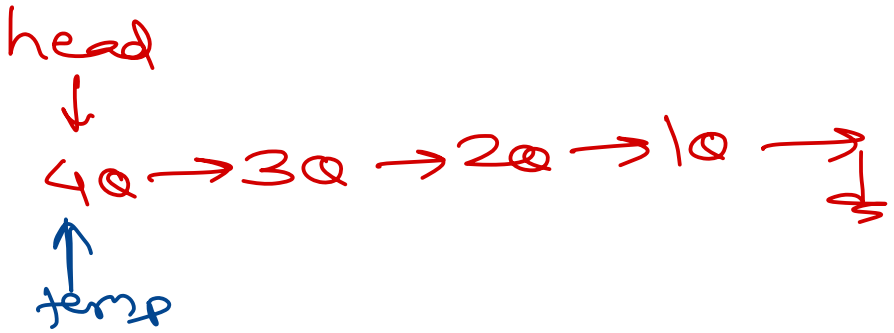
Data Structure & Algorithms

Nilesh Ghule

Linked List – Competitive programming

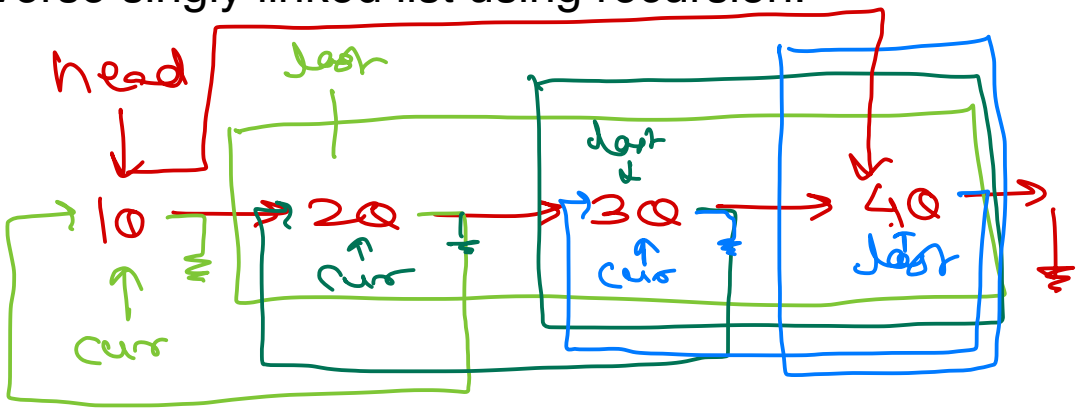
- Reverse singly linked list.



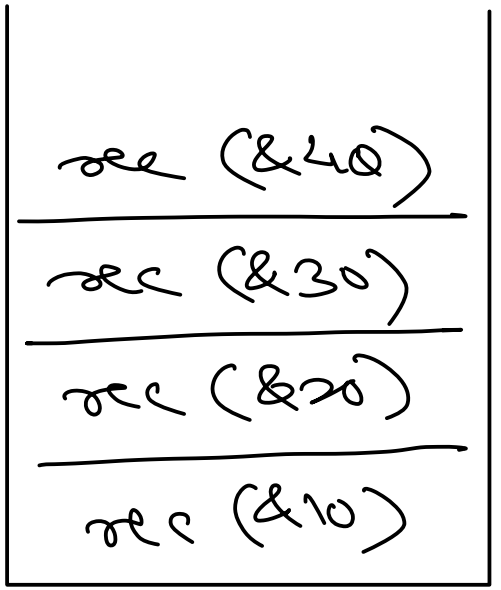

```
oldhead = head;
head = null;
while(oldhead != null) {
    temp = oldhead;
    oldhead = oldhead.next;

    temp.next = head;
    head = temp;
}
```

Linked List – Competitive programming

- Reverse singly linked list using recursion.

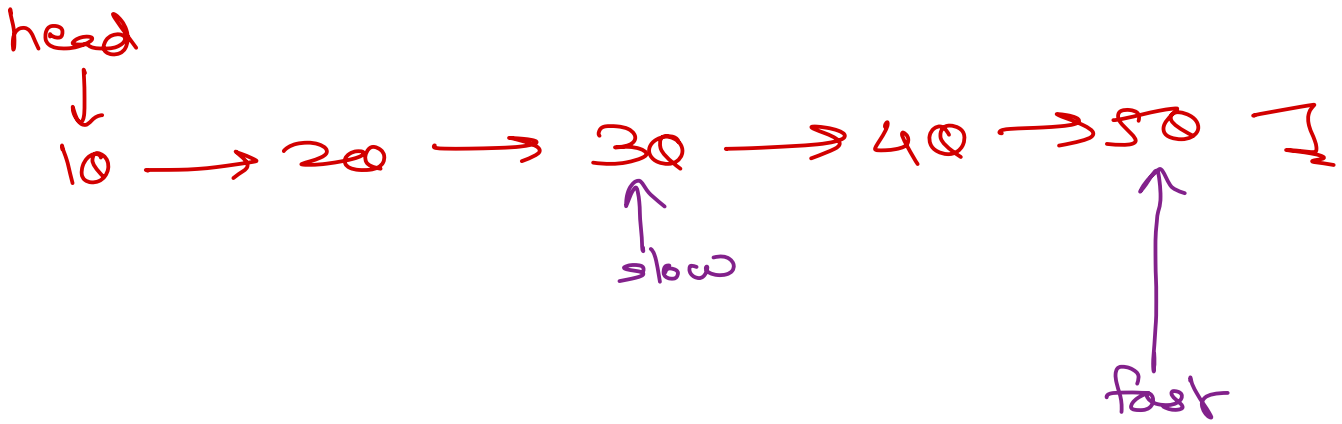



```
last Node recReverse(Node cur) {
    if (cur.next == null) {
        head = cur;
        return cur;
    }
    last = recReverse(cur.next);
    last.next = cur;
    cur.next = null;
    return cur;
}
```



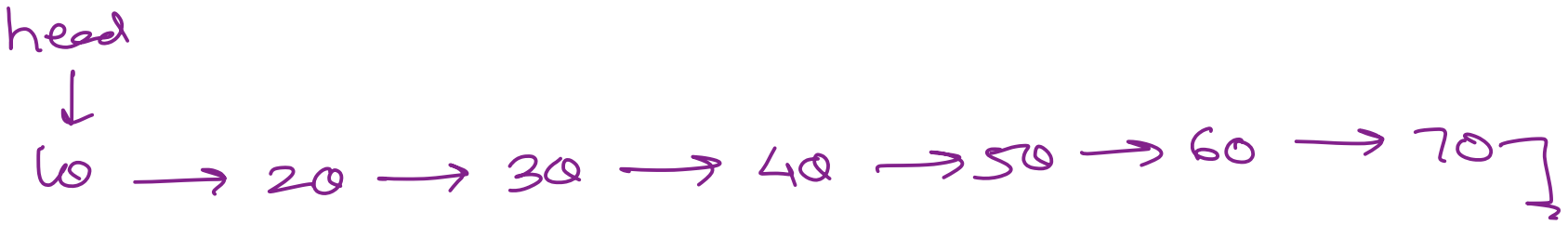
Linked List – Competitive programming

- Find middle of singly linear linked list.

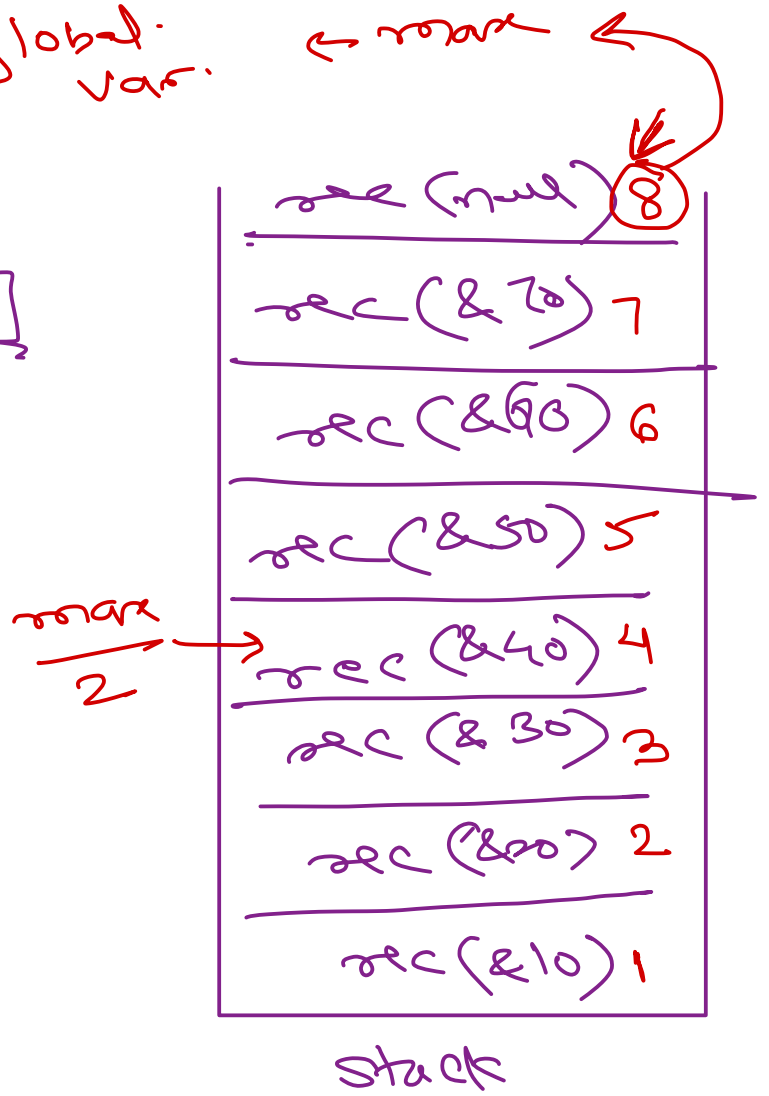


Linked List – Competitive programming

- Find middle of singly linear linked list using single pointer.

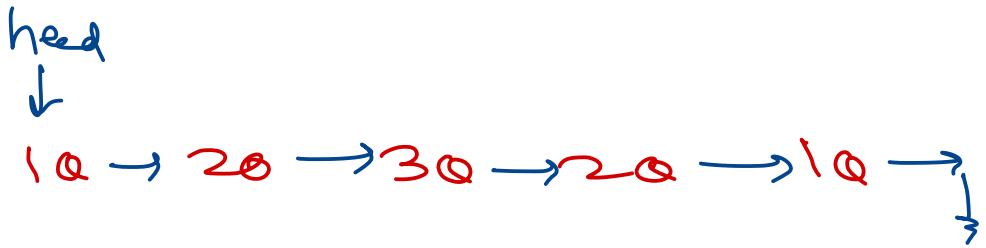


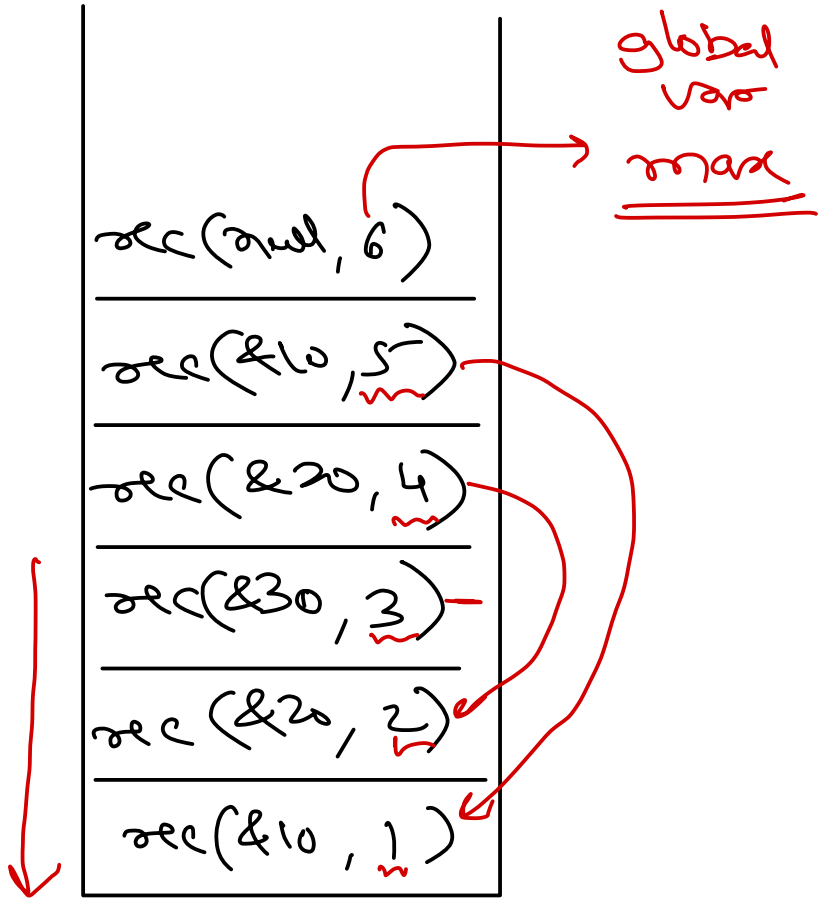

```
int rec FindMid(Node cur, int count) {
    if(cur == null) {
        max = count;
        return 0;
    }
    mid = rec FindMid(cur.next, count+1);
    if(count == max/2)
        mid = cur.data;
    return mid;
}
```



Linked List – Competitive programming

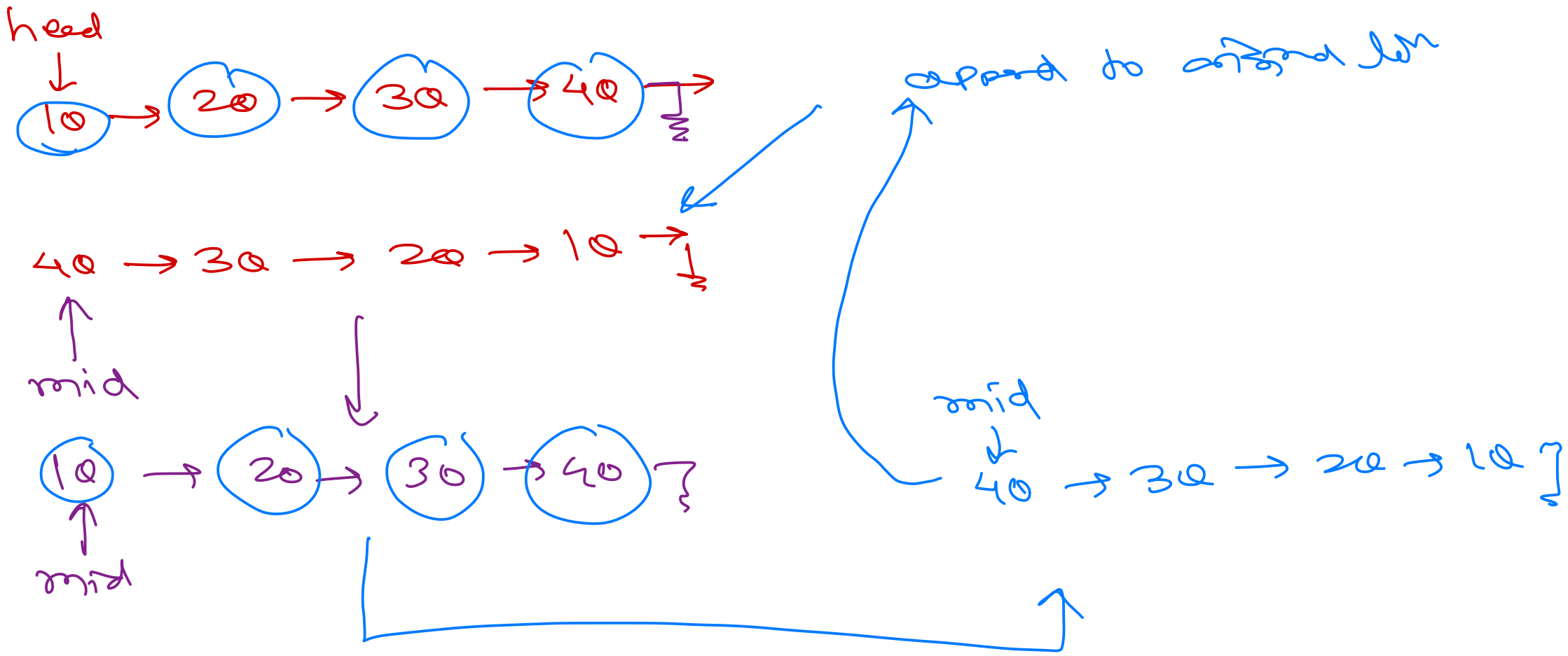
- Check if linked list is palindrome.





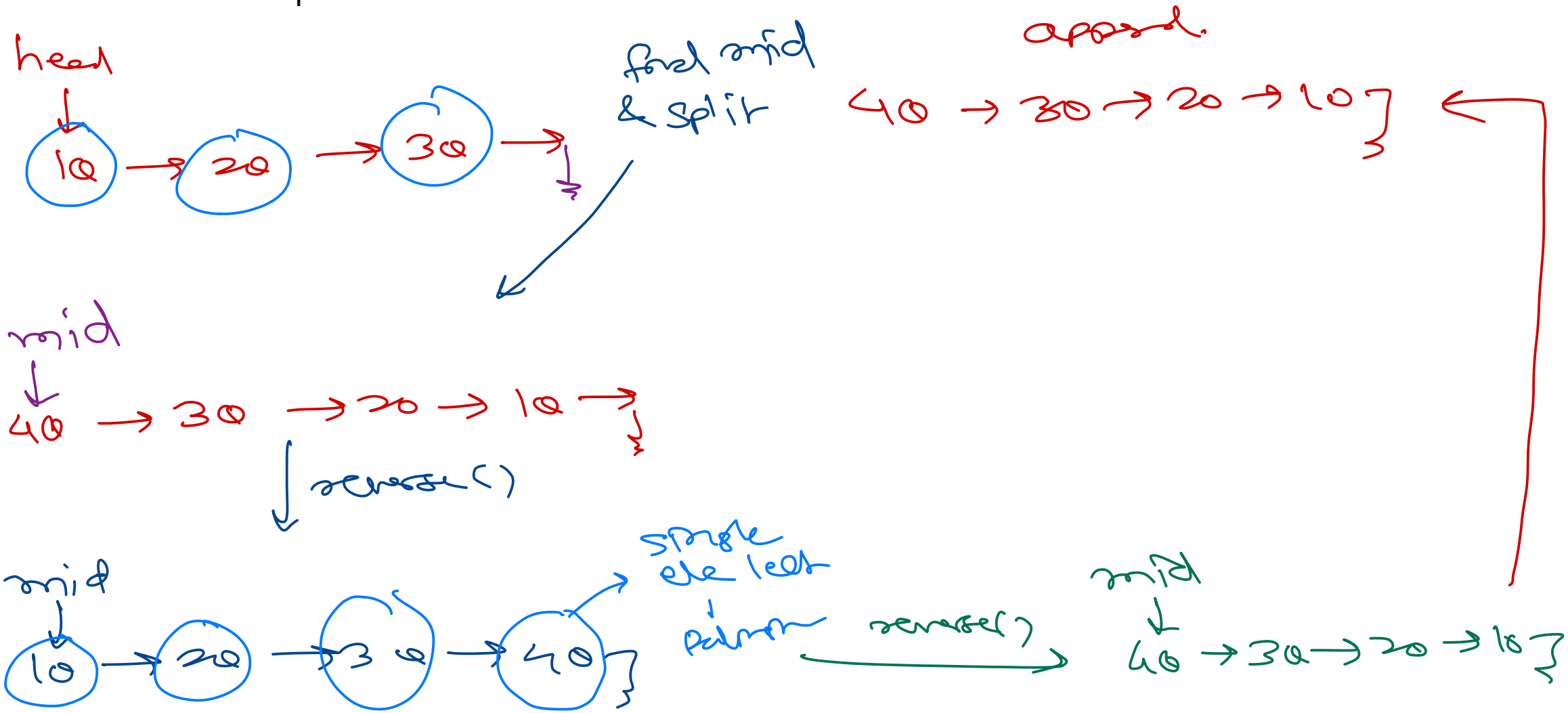
Linked List – Competitive programming

- Check if linked list is palindrome.



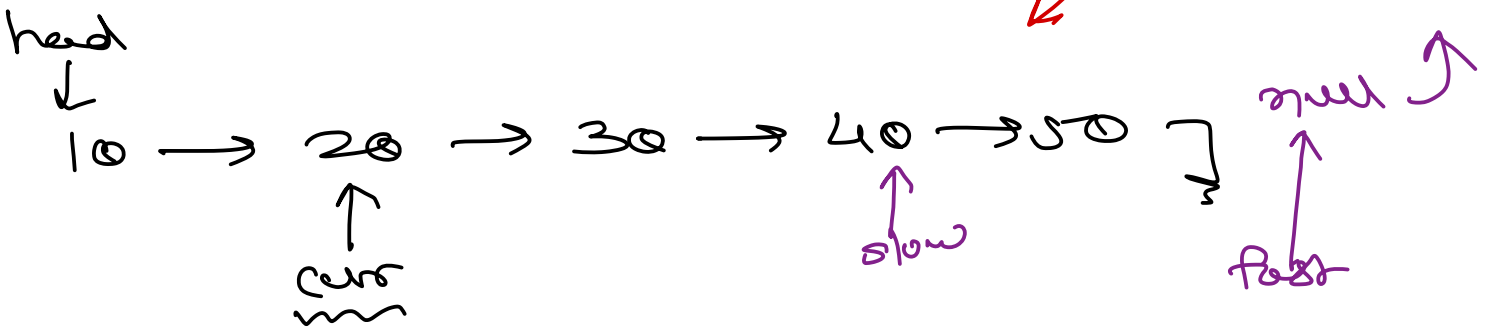
Linked List – Competitive programming

- Check if linked list is palindrome.



Linked List – Competitive programming

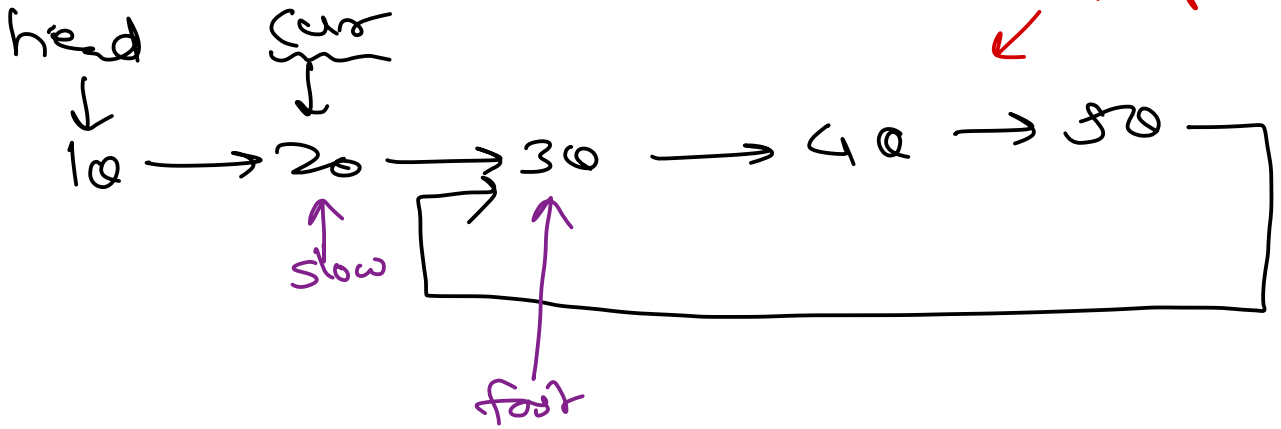
- Check if linked list contains a loop.



no loop = false

infinite loop

```
while (cur != null) {
    cur = cur.next;
    if (cur == null)
        return false;
}
```



loop = true

```
slow = cur;
fast = slow.next;
while(slow != fast) {
    if(fast == null ||
       fast.next == null)
        return false;
    slow = slow.next;
    fast = fast.next.next;
}
return true; // loop
```

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>