mergeSort

↓

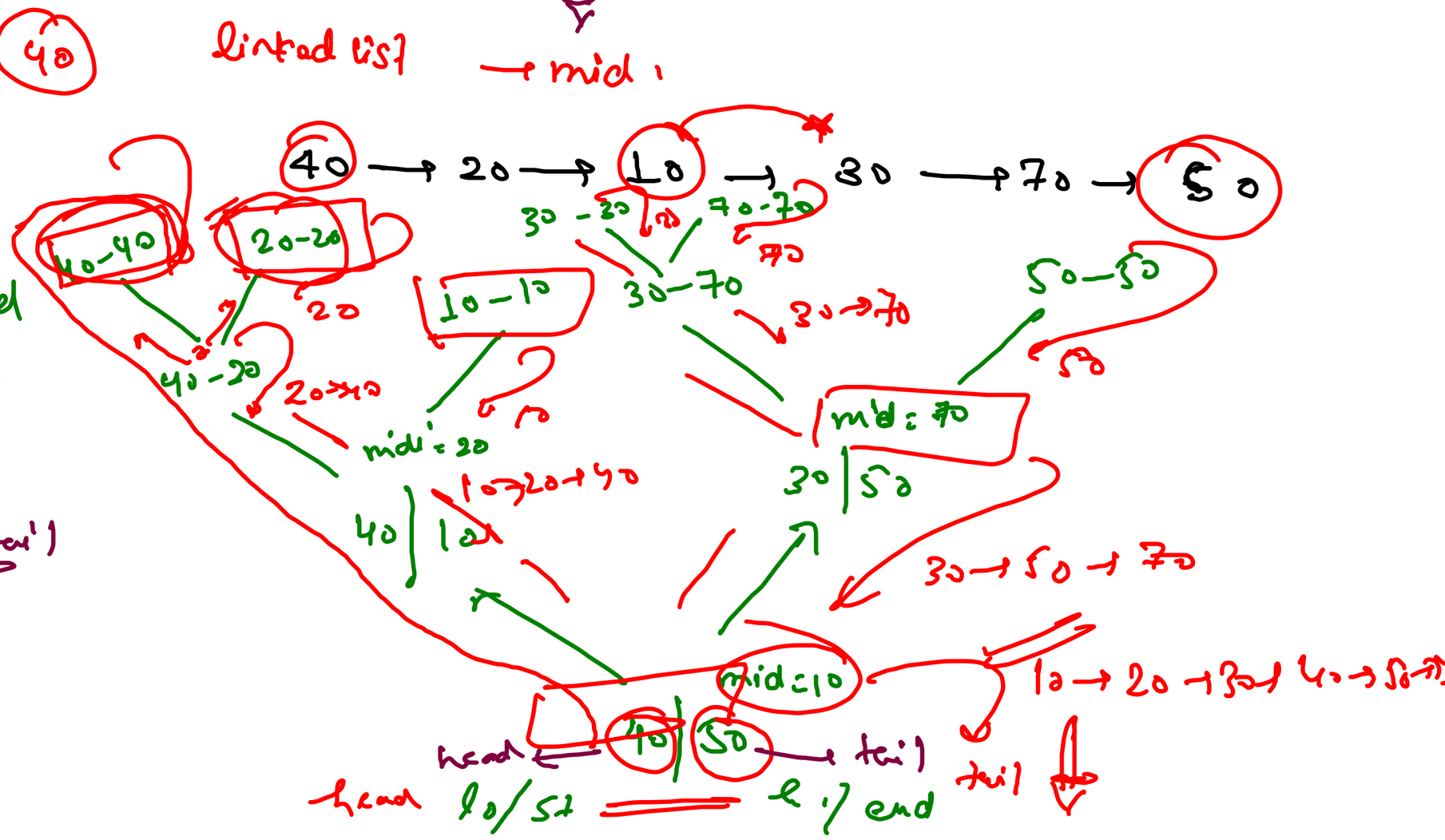
mid → array

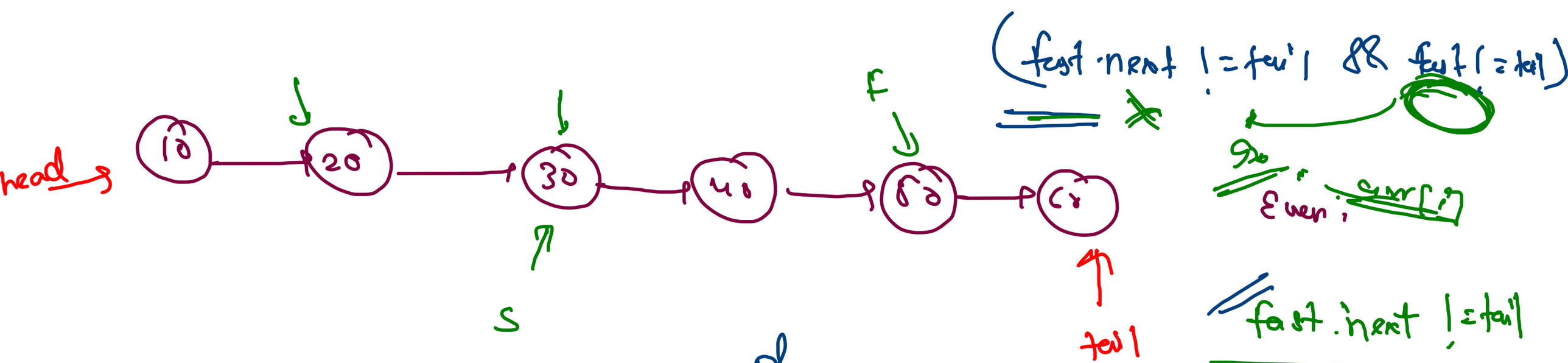
$\left(\frac{lo + hi}{2}\right)$

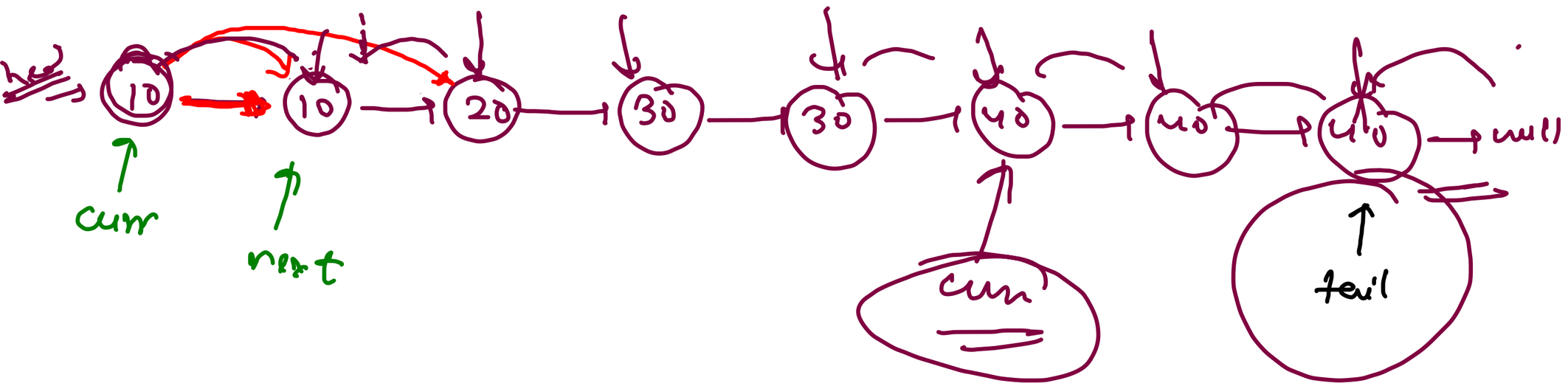
recursion

Merging of two sorted Linked list

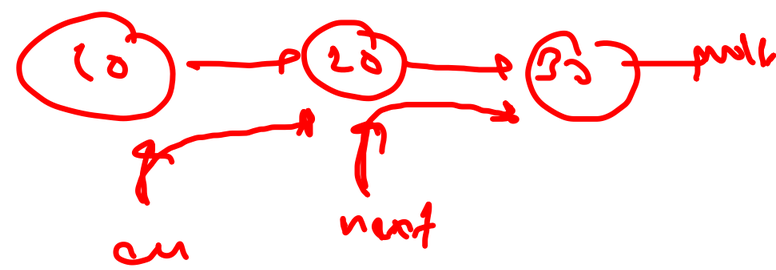
Base case = head == tail



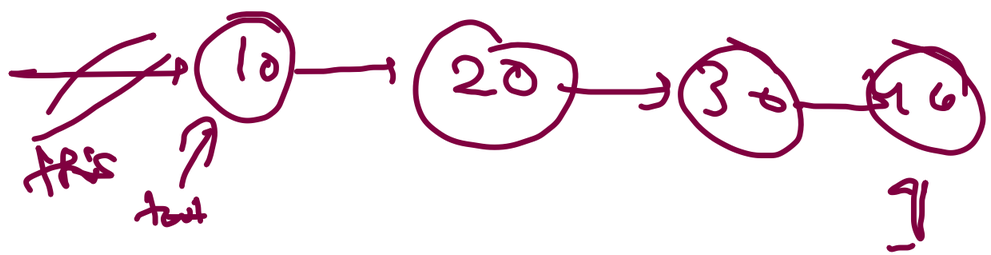

head
10
20
30
40
50
60
s
f
tail
(fast.next != tail && fast != tail)
Even
mid
head
10
20
30
40
50
60
70
tail
fast != tail
if (fast != tail && fast.next != tail)
fast != tail



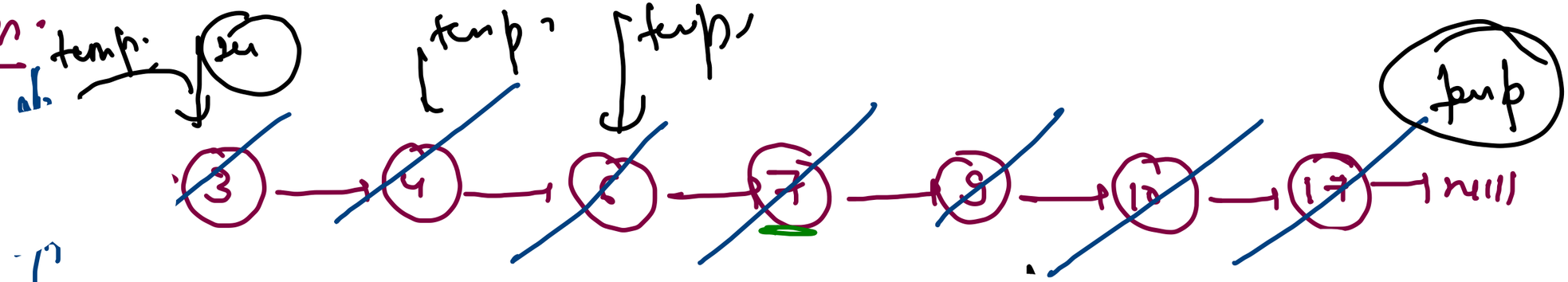

```
curr.next = next;
if( curr.data == next.data) {
    next = next.next;
} else{
    curr = curr.next;
    next = next.next;
}
```

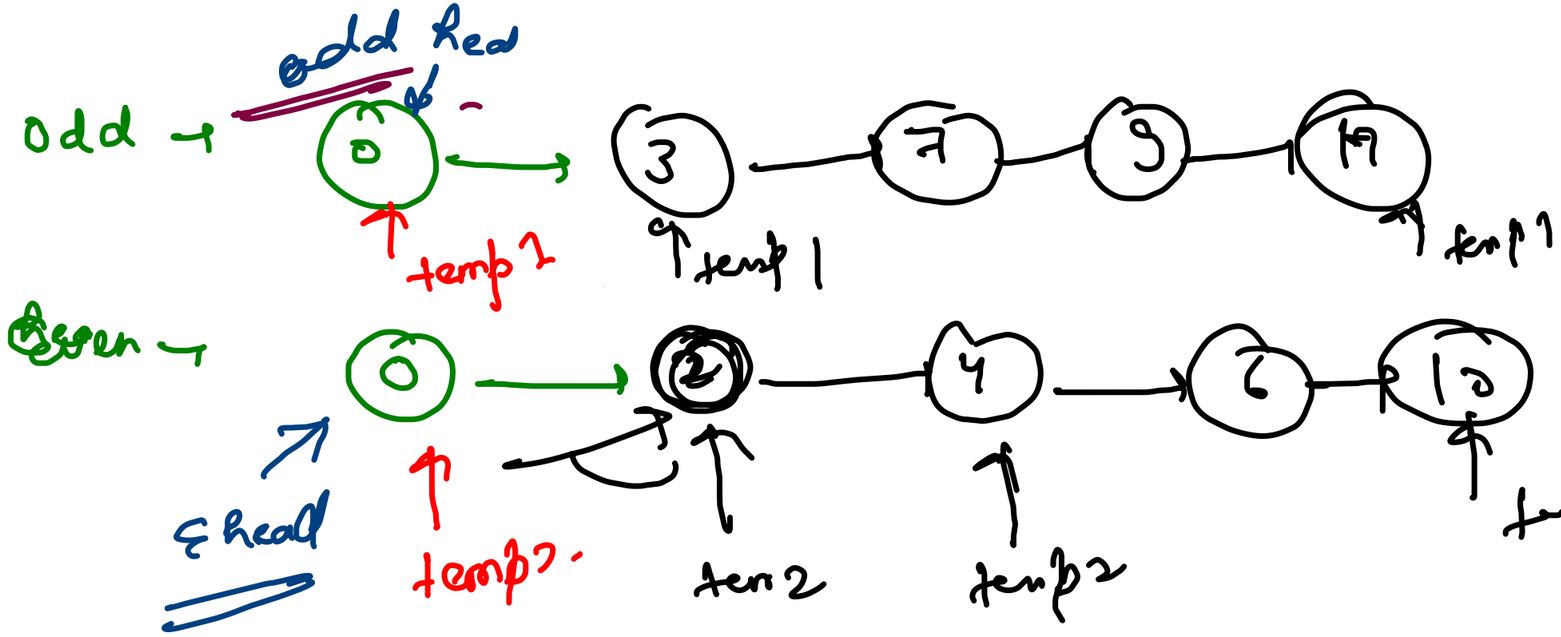


Linked list



odd Even :





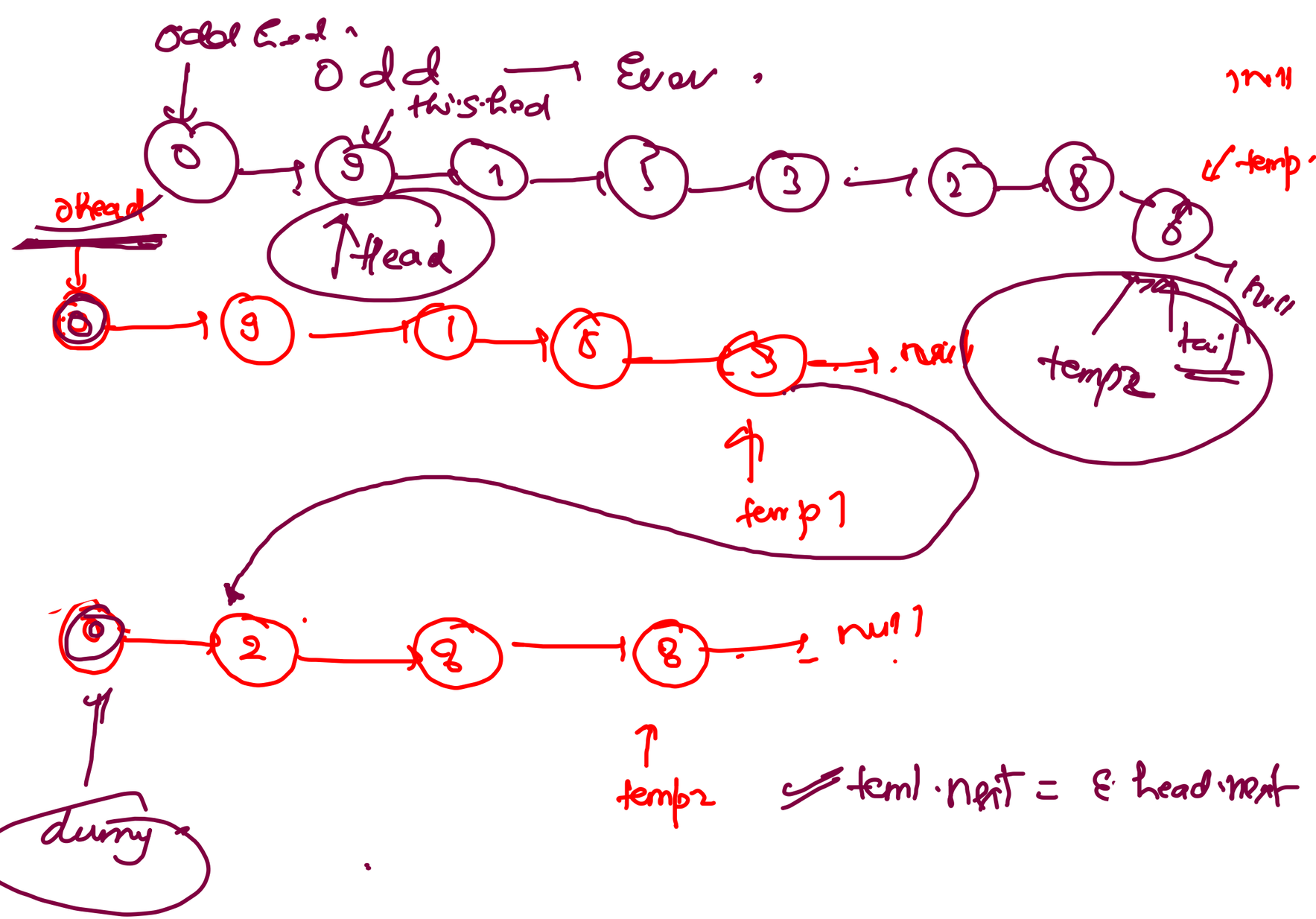

```
if( data % 2 == 0 ) {
    temp2.2.next = temp;
    temp = temp.next
    temp2 = temp2.next;
} else {
    temp1.next = temp;
    temp = temp.next;
    temp1 = temp2.next
}
```

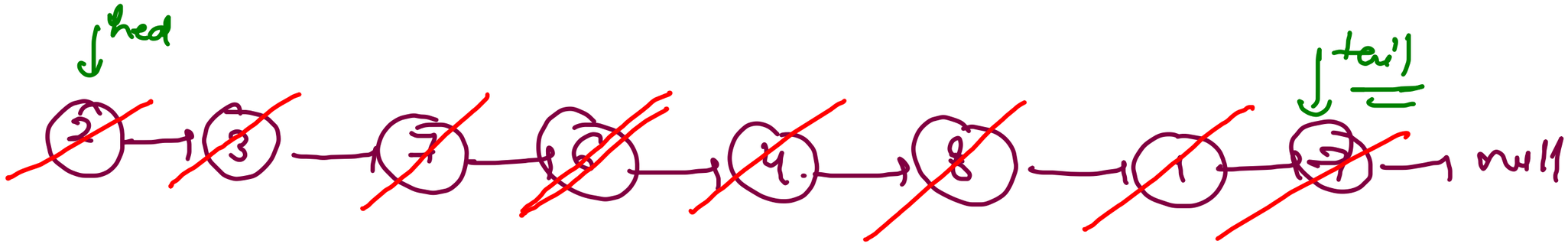
~~odd~~

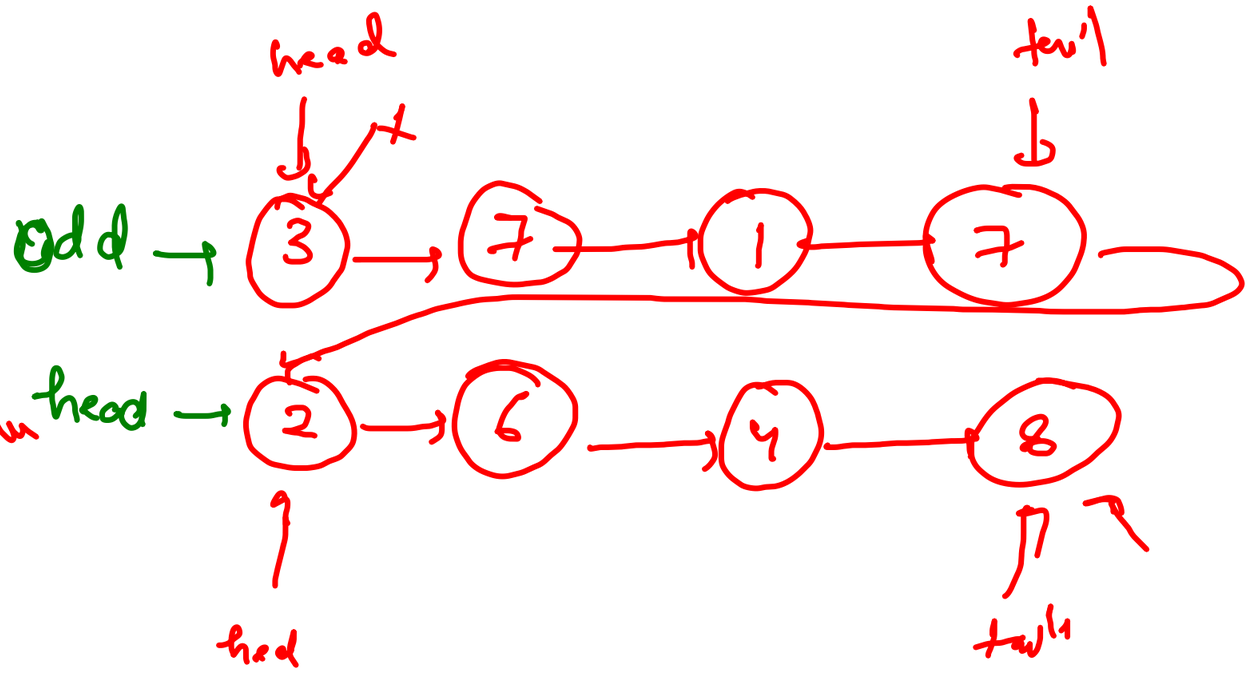
Q.

```
temp1.next = Eread.next;
this.head = odd Read.next;
this.tail = temp2;
```

```java
public void oddEven() {
    Node temp1 = new Node();
    Node ohead = temp1;

    Node temp2 = new Node();
    Node ehead = temp2;

    Node temp = head;

    while(temp != null) {
        int data = temp.data;

        if(data % 2 == 0) {
            // even
            temp2.next = temp;
            temp2 = temp2.next;
        } else {
            // odd
            temp1.next = temp;
            temp1 = temp1.next;
        }
        temp = temp.next;
    }

    temp1.next = ehead.next;
    this.head = ohead.next;
    this.tail = temp2;
```



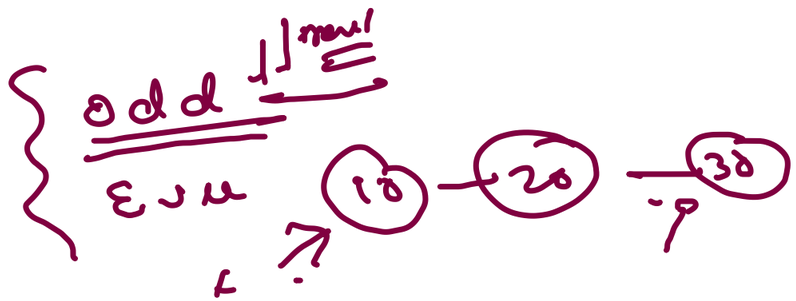


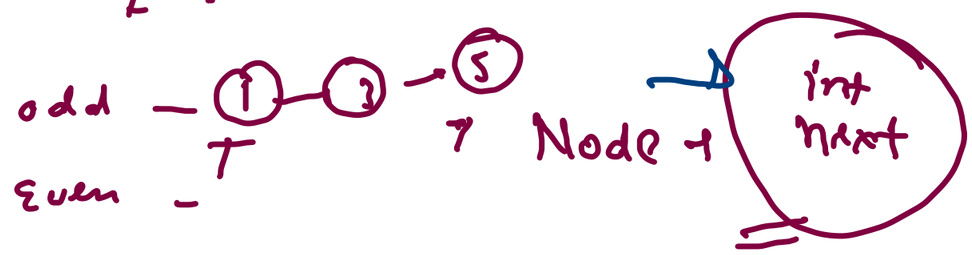


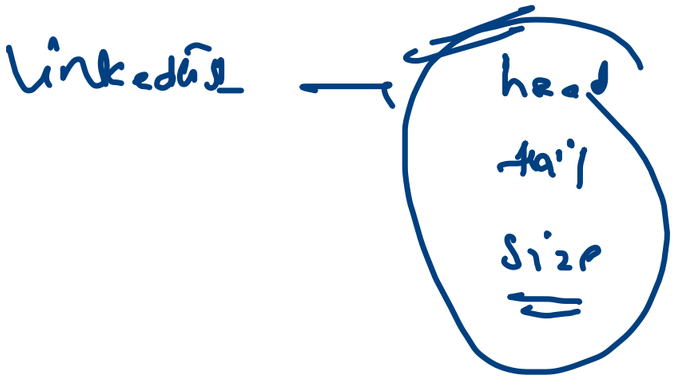
odd.tail.next = Even.head;

this.head = odd.head;

this.tail = Even.tail



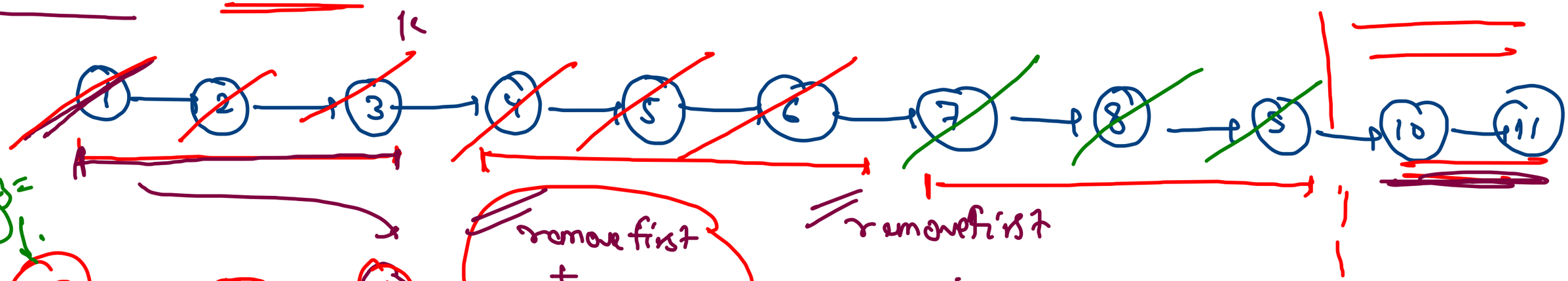


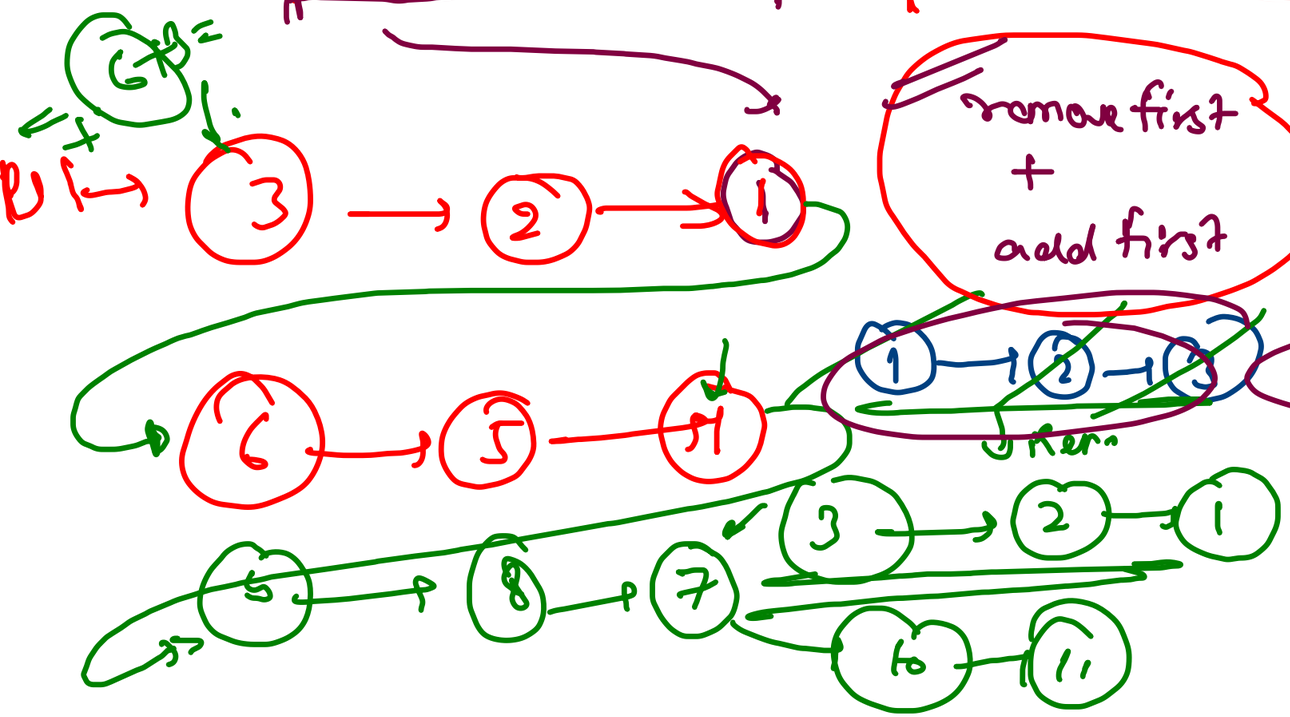


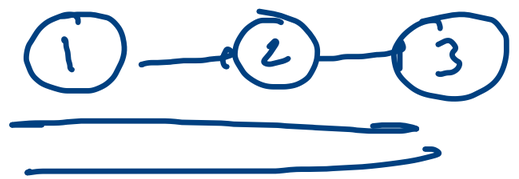
k-Reverse.

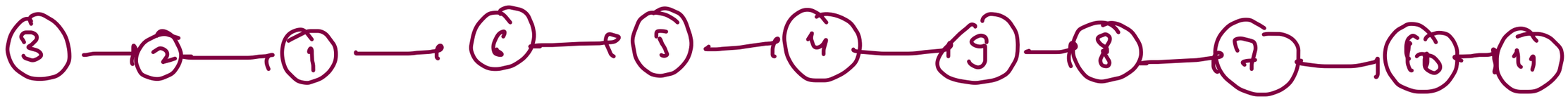
k = 3

Size < 11









k = 3

removeFirst

ptail.next = curr.head

ptail = curr.tail;

p.size = curr.size;

p.head

p.tail

prev = null

prev

curr

ptail = curr.tail

Remove first

if(prev == null) {

prev = curr;

} else {

ptail.next = curr.head

ptail = curr.tail;

p.size = curr.size;

head

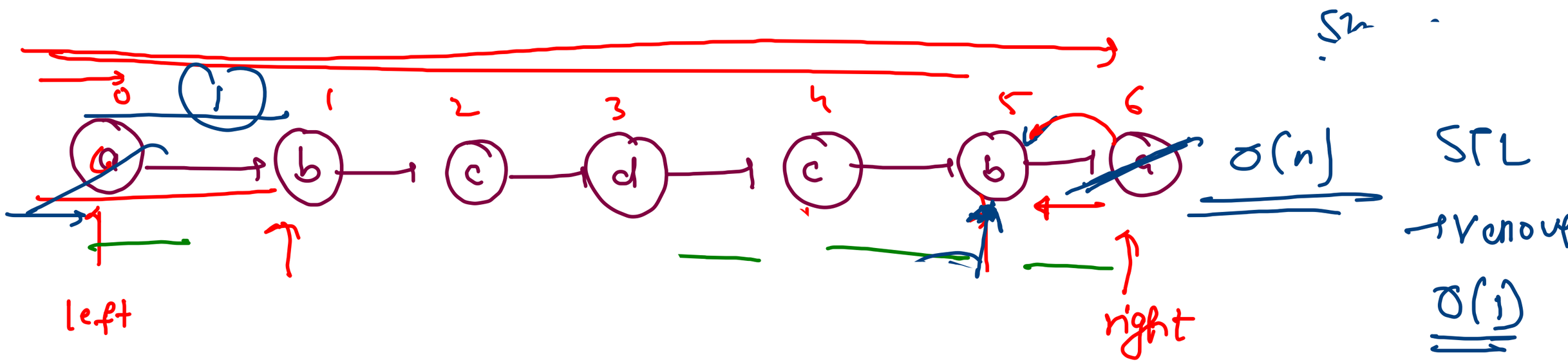
tail

size

curr

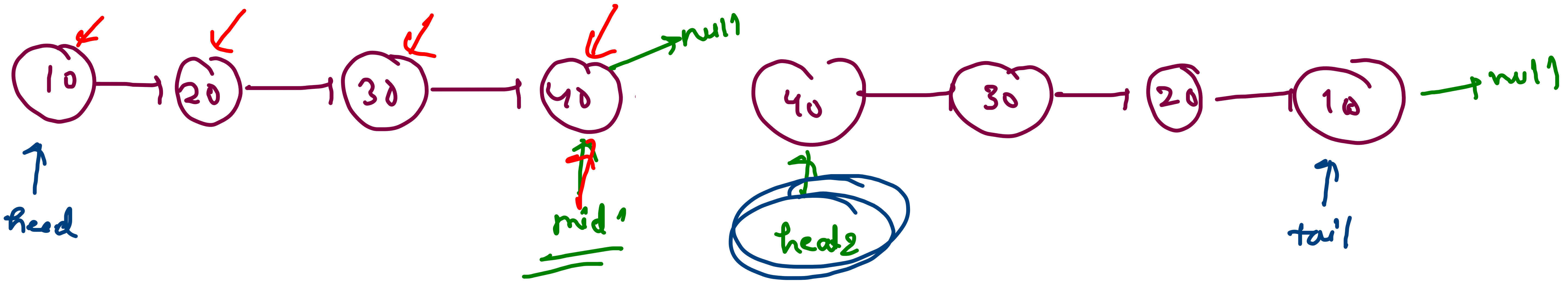
prev

(1)




```java
// write your code here
int left = 0;
int right = this.size - 1;

boolean isPalin = true;
while(left < right) {
    int ldata = this.getAt(left);
    int rdata = this.getAt(right);

    if(ldata != rdata) {
        isPalin = false;
        break;
    }

    left++;
    right--;
}

return isPalin;
}
```

Even.

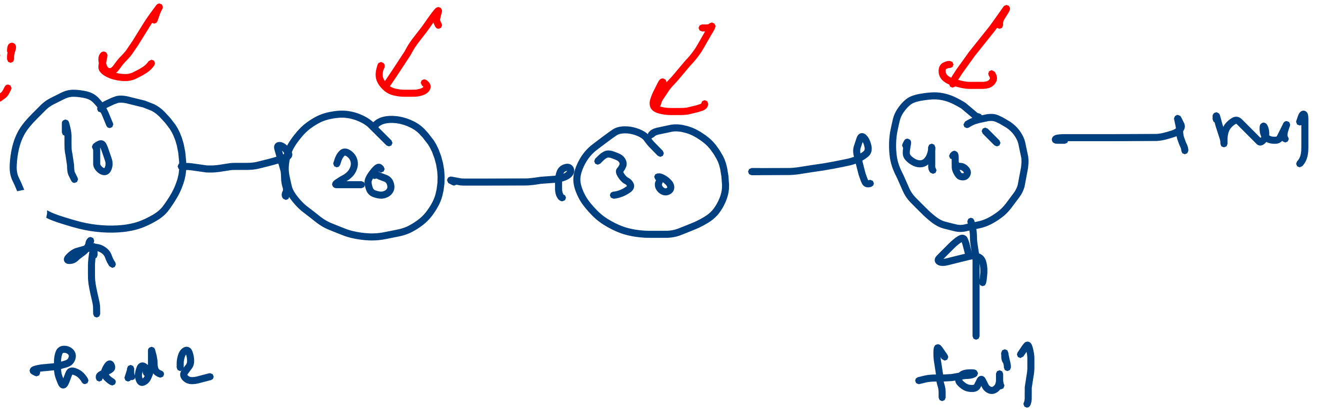


mid

Equal

isPalindrome = False

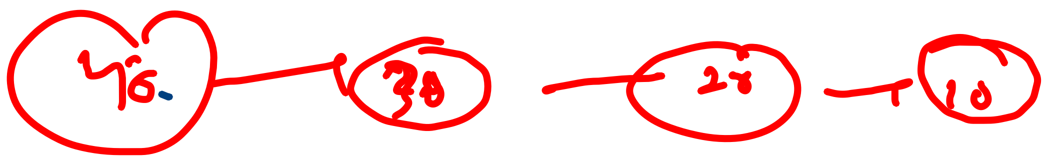
return false



head2 = mid.next;

mid.next = null

reverse(head2) → return type → Node

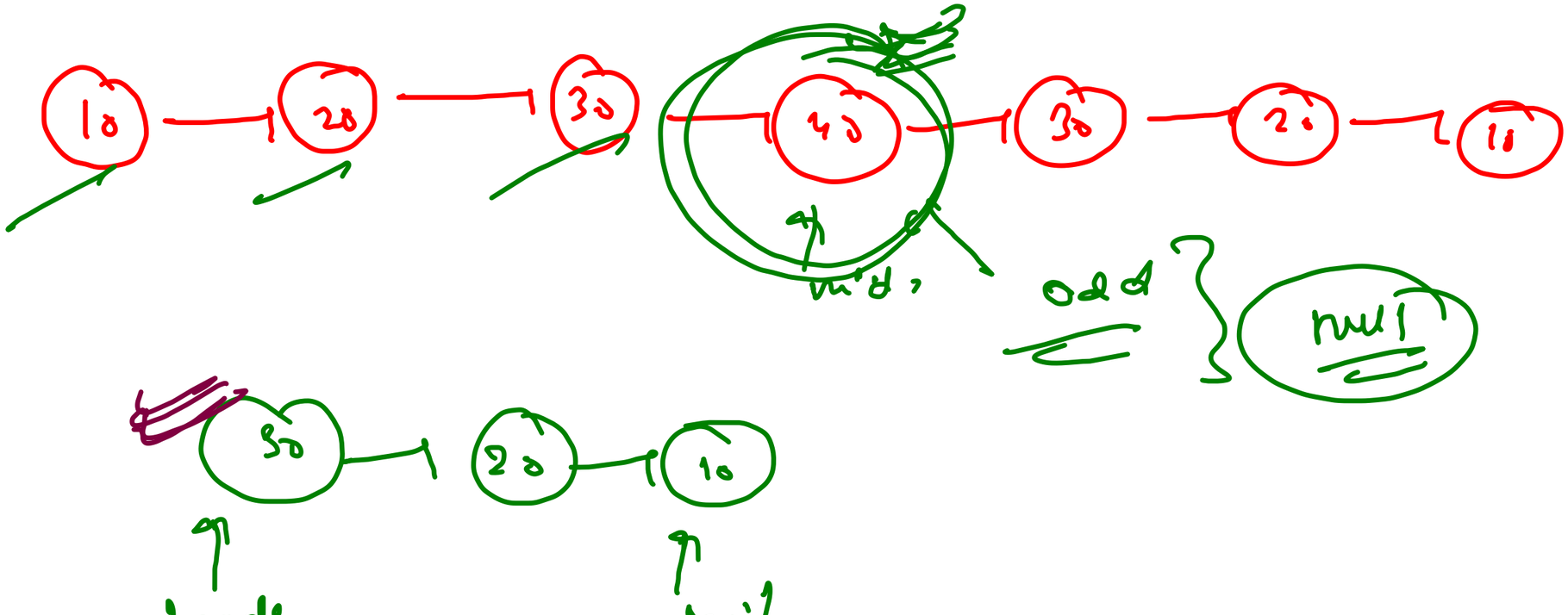


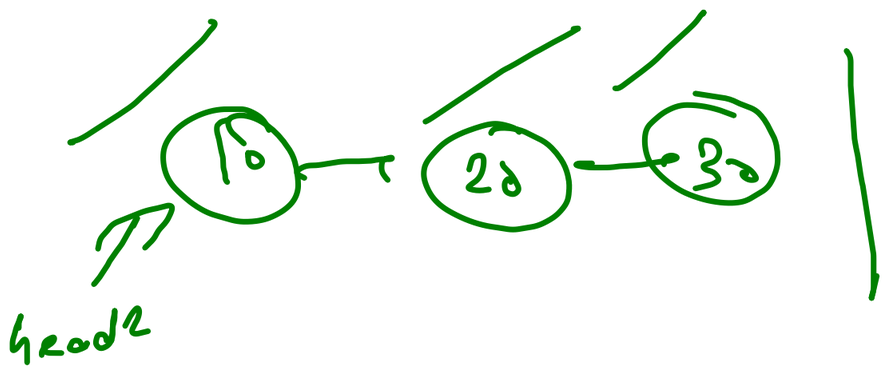
check for palindrome;

reverse fun → pointer reverse.
→ Return the Node

reverse(head2);

mid.next = head2;


10
20
30
40
30
20
11
odd
null
30
20
10
head
tail


10
20
30
head2