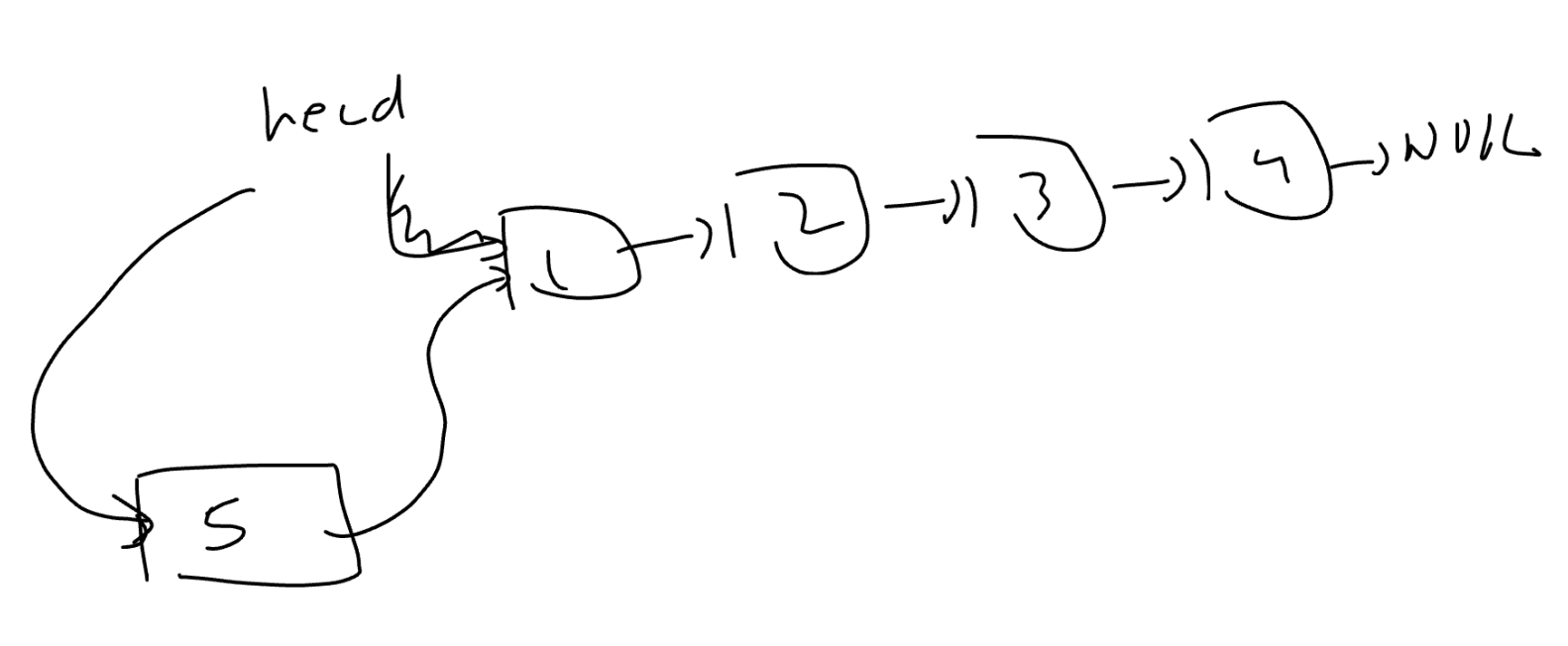
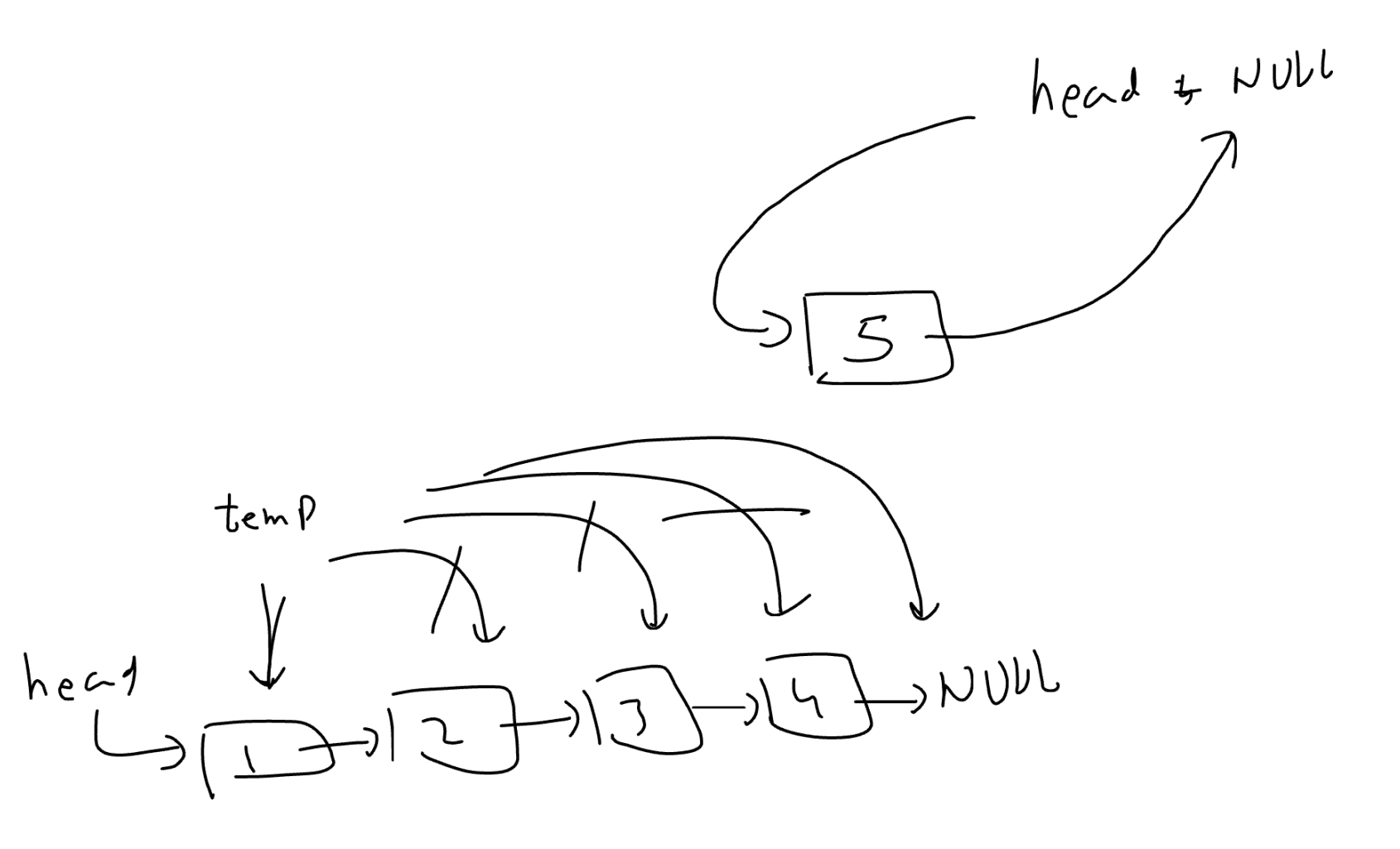
head
↳ [1] → [2] → [3] → [4] → NULL.

head
↳ [5] → [1] → [2] → [3] → [4] → NULL.

newNode = new Node(val);
newNode → next = head.
head = newNode.





1,2,3,4

1 2 3 4

1 2 3 4

[while (head != NULL)
print head
head = head → next]

[5]

temp
↓

4 → next = NULL.

I/P: 1 → 2 → 3 → 4 → NULL

O/P: 1 → 2 → 3 → 4 → 5 → NULL.

[→ newNode = new node(val)
→ temp → next = newNode.

NULL →

NPE.

pos = 50

temp [6]

I/P: 1 → 2 → 3 → 4 → 5 → NULL.   pos = 2, val = 6

O/P: 1 → 2 → 6 → 3 → 4 → 5 → NULL.   min(pos, n).

1) pos − 1 hops to reach 2
2) Create a new node.
3) newNode → next = temp → next
4) temp → next = new Node.

Edge_case:
pos > n
pos = 0
head == NULL.

temp
↓
1 → 2 → 3 → 4 → 5 → NULL [6]

toDel
head
7 pos = 4, val = 7.

I/P: 0 → 1 → 2 → 3 → 4 → 5 → 6 → 10.

O/P: head ↳ 1 → 2 → 3 → 7 → 4 → 5 → 6 → 10.

toDel = head
head = head → next.
delete(toDel).

1 → ~~2 → 3~~ 3

temp toDel
↓ ↓

I/P: 1 → 2 → 3 → 4 → 5 → NULL.

O/P: 1 → 2 → 3 → 4 → NULL.

temp → next → next == NULL.

toDel = temp → next
temp → next = NULL.
delete(toDel).

temp toDel
↓ ↓

I/P: 1 → 2 → 3 → 7 → 4 → 5 → 6 → NULL, pos = 3

O/P: 1 → 2 → 3 → 4 → 5 → 6 → NULL.

head == NULL || pos = 0

1) pos − 1 hops to reach 3
2) toDel = temp → next
3) temp → next = temp → next → next
4) delete(toDel).

I/P: 1 → 2 → 3 → 4 → 5 → 6 → 7 → NULL.

O/P: 7 → 6 → 5 → 4 → 3 → 2 → 1 → NULL.

1 → 2 → 3 → 4 → 5 → 6 → 7 → NULL.
prev curr temp

← 1 ← 2 ← 3 ← 4   5 → 6 → 7 → NULL.
prev curr

prev curr
← 1 ← 2 ← 3 ← 4   5 → 6 → 7 → NULL

1) temp = curr → next
2) curr → next = prev
3) prev = curr
4) curr = temp

curr = temp

prev curr t t t t t
NULL ← 1 → 2 → 3 → 4 → 5 → 6 → 7 → NULL.
p c

Terminal condition to stop = curr != NULL.

New head = prev.

TC: O(n)
AS: O(1).

prev curr t
NULL ← 1 → NULL.
P C

---

slow
↓
I/P: 1 → 2 → 3 → 4 → 5 → NULL.
fast↑
O/P: 3.

5/2 hops.
OR
n/2 hops.

Runner's Technique.

In 1 iteration:
slow moves 1 stop : slow = slow → next
and
fast moves 2 stops : fast = fast → next → next

s s s
1 → 2 → 3 → 4 → 5 → NULL.
f f f

[fast → next == NULL] ODD

s s s s
1 → 2 → 3 → 4 → 5 → 6 → 7 → NULL.
f f f

s s s s
1 → 2 → 3 → 4 → 5 → 6 → NULL.   O/P = 4
f f f

[fast == NULL] Even.

~~1 → 2 → 3 → 4 → 5 → 6 → 7 → 8~~ → 9 → 10 → NULL.

0 1 2 3 4
1 → 2 → 3 → 4 → N.
f

0 1 2 3 4 5
1 → 2 → 3 → 4 → 5 → N.

(fast != NULL && fast → next != NULL)

a b
for (int i=0; i<n; i++)
sum1 += a[i]
sum2 += b[i]

for (int i=0; i<n; i++)
sum1 += a[i]
for (int i=0; i<n; i++)
sum2 += b[i].

s s s
1 → 2 → 3 → 4 → 5 → 6 → NULL.
f f f

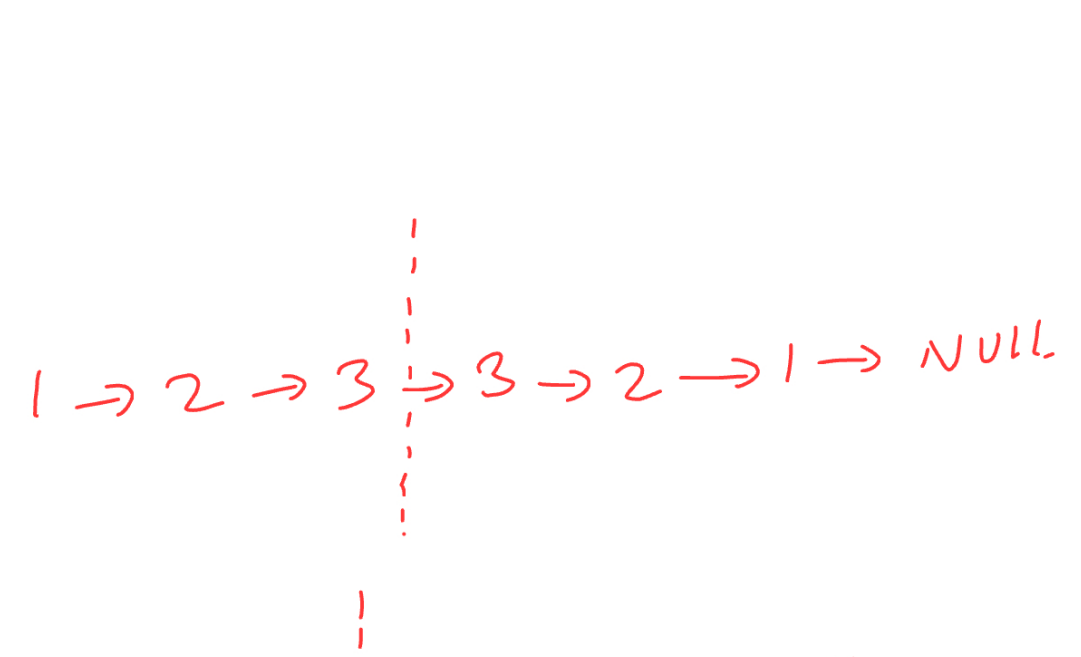
[fast → next != NULL && fast → next → next != NULL]

1a → 2a → 3a → 4 → 3b → 2b → 1b → NULL.

NULL ⤸ return 1a
1b ⤸ return 1a → next = 2a
2b ⤸ return 2a → next = 3a
3b ⤸ 3a → next = 4
4 ⤸ 4 → next = 3b
3a ⤸ 3b → next = 2b
2a ⤸ 2b → next = 1b
1a ⤸

d
c
b
a

TC: O(n)
AS: O(n).



```
Node* isPalindromeRecUtil(Node* curr, Node* head, bool &result) {
    if (curr == NULL) {
        return head;
    }

    Node* temp = isPalindromeRecUtil(curr->next, head, result);
    if (curr->data != temp->data) {
        result = false;
    }

    return temp->next;
}
```

head
↳ 1a → 2a → 3 → 2b → 1b → N.

c = NULL ⤸ 1a
c = 1b ⤸ 2a
c = 2b ⤸ 3
c = 3 ⤸ 2b
c = 2a ⤸ 1b
c = 1a

1 → 2 → 3 → 3 → 2 → 1 → NULL

1 → 2 → 3 → 2 → 1 → NULL.

mid head2
head ↓ ↓
↳ 1 → 2 → 3 → 2 → 1 → NULL.
↳ NULL

[head2 = mid → next
mid → next = NULL.

h s f
↳ 1 → 2 → 3 → N.
h2
↳ 1 → 2 → N

mid head2
head ↓ ↓
↳ 1 → 2 → 3 → 3 → 2 → 1 → NULL
↳ NULL

h → 1 → 2 → 3 → NULL
h2 ↳ 1 → 2 → 3 → NULL

TC: O(n)
AS: O(1).