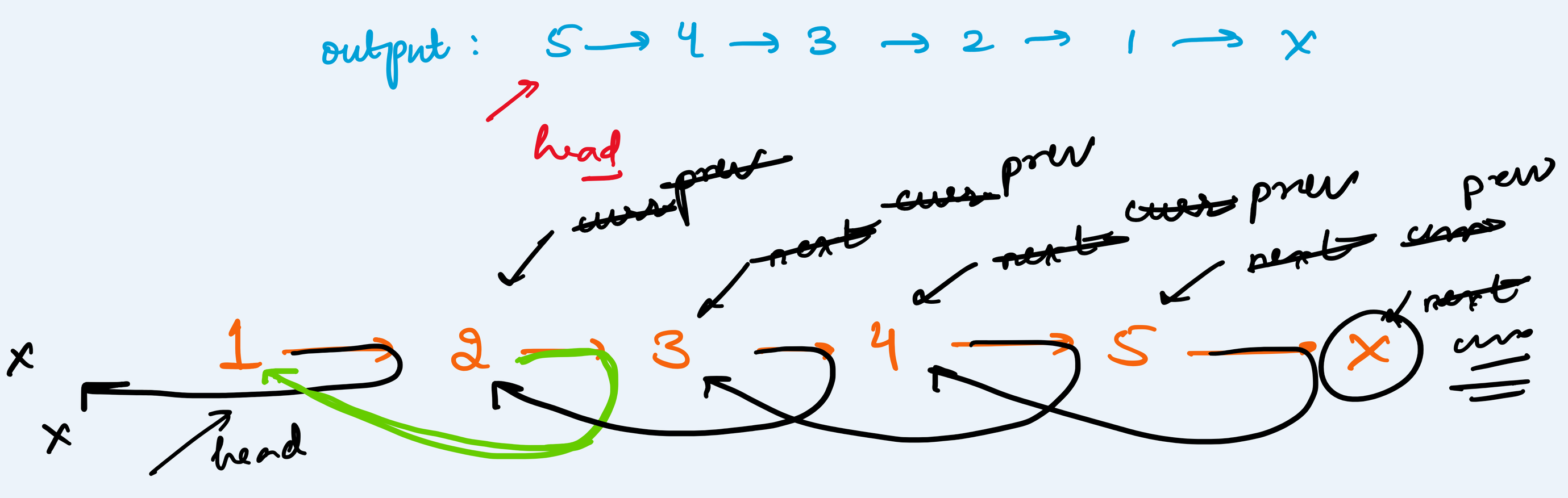
Wednesday, 22 September 2021 5:28 PM

Q1 Given a singly linked list, reverse the linked list.

Eg: head → 1 → 2 → 3 → 4 → 5 → x

output: 5 → 4 → 3 → 2 → 1 → x (head)




```
while ( curr != null) {
    next = curr.next
 →  curr.next = prev;
    prev = curr;
    curr = next;

}
```

Q Given a non-empty linked list, with a head node, return the middle of a linked list.

→ If 2 middle elements, return the second one.

Eg: 1 → 2 → 3 → 4 → 5 → x
ans → (3)

Eg: 1 → 2 → 3 → 4 → 5 → 6 → x
ans → (4)

Approach (1): → Find the size of LL → (n)

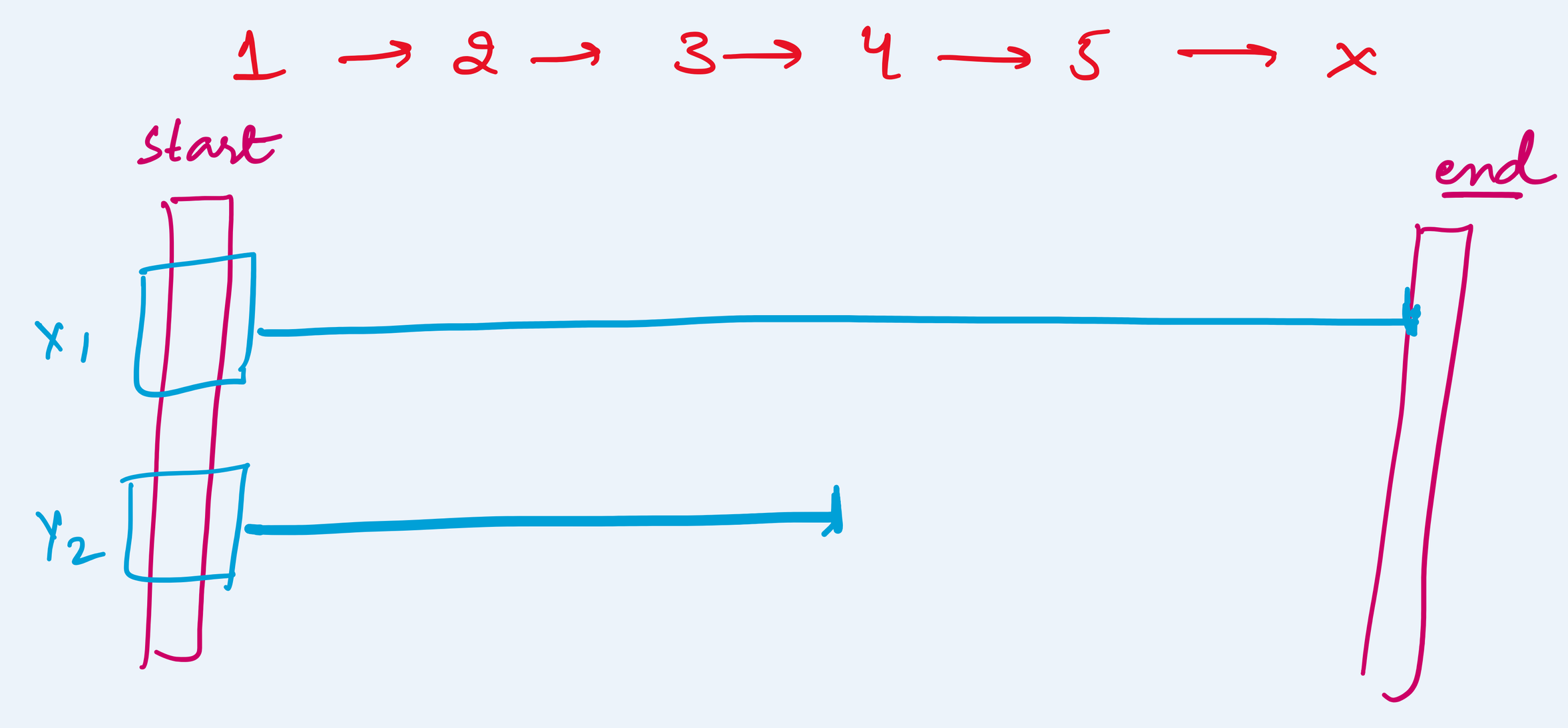
→ → If (size == even) → $\frac{n}{2} + 1$

→ If (size == odd) → $\frac{n}{2}$

```
Node temp = head;
  count = 0;
while ( temp != null) {
        temp = temp.next;
        count ++;
}
```
→ (1)

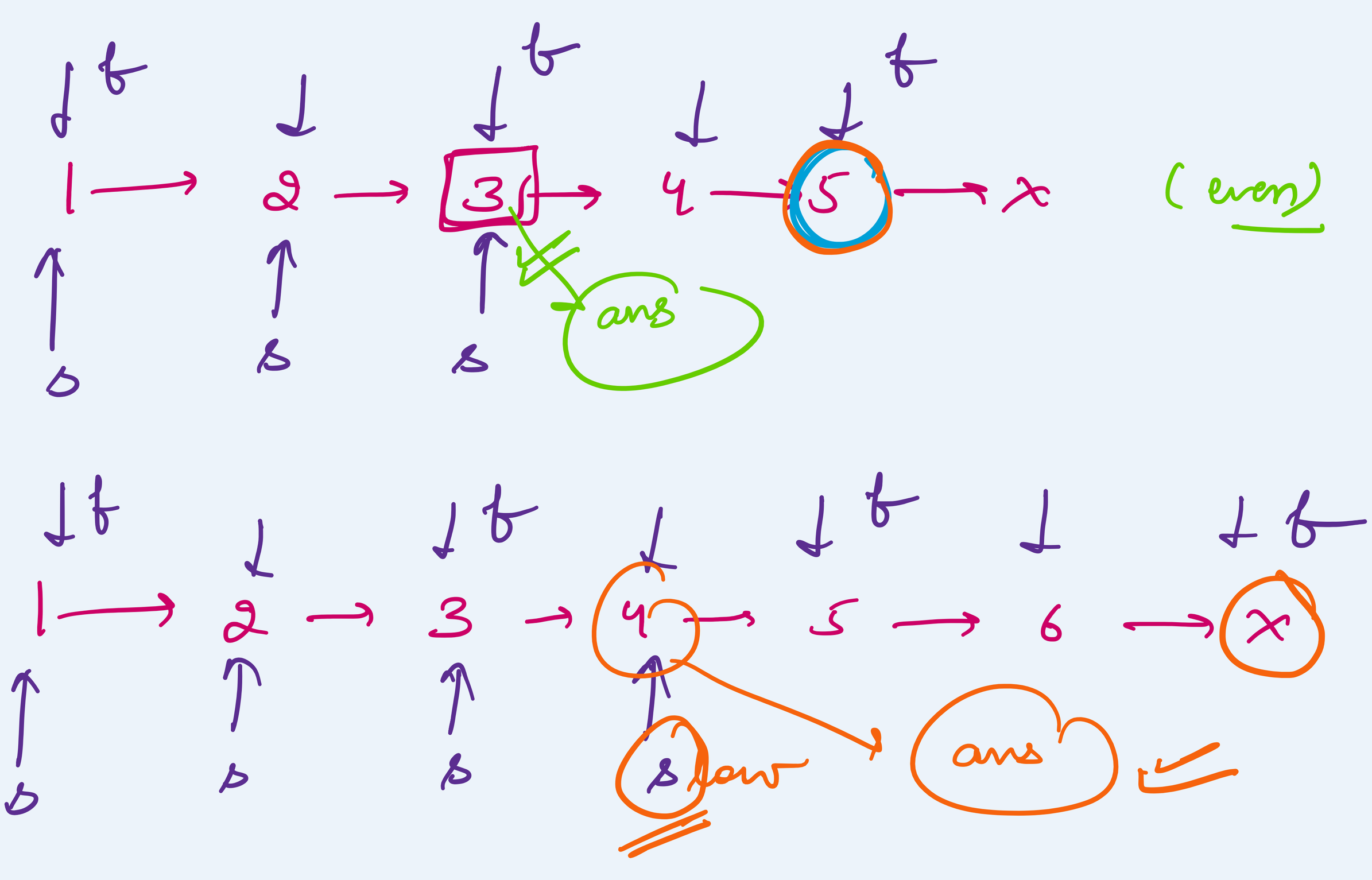
```
Node temp = head;
int itr = 0;
if (count % 2 == 0) {
        itr = count/2 + 1;
} else {
        itr = count/2;
}
for (int i=0; i< itr; i++) {
        temp = temp.next;
}
```
→ (2)

Approach (2):



speed (x_1) = 2 * speed (x_2)

t = 0



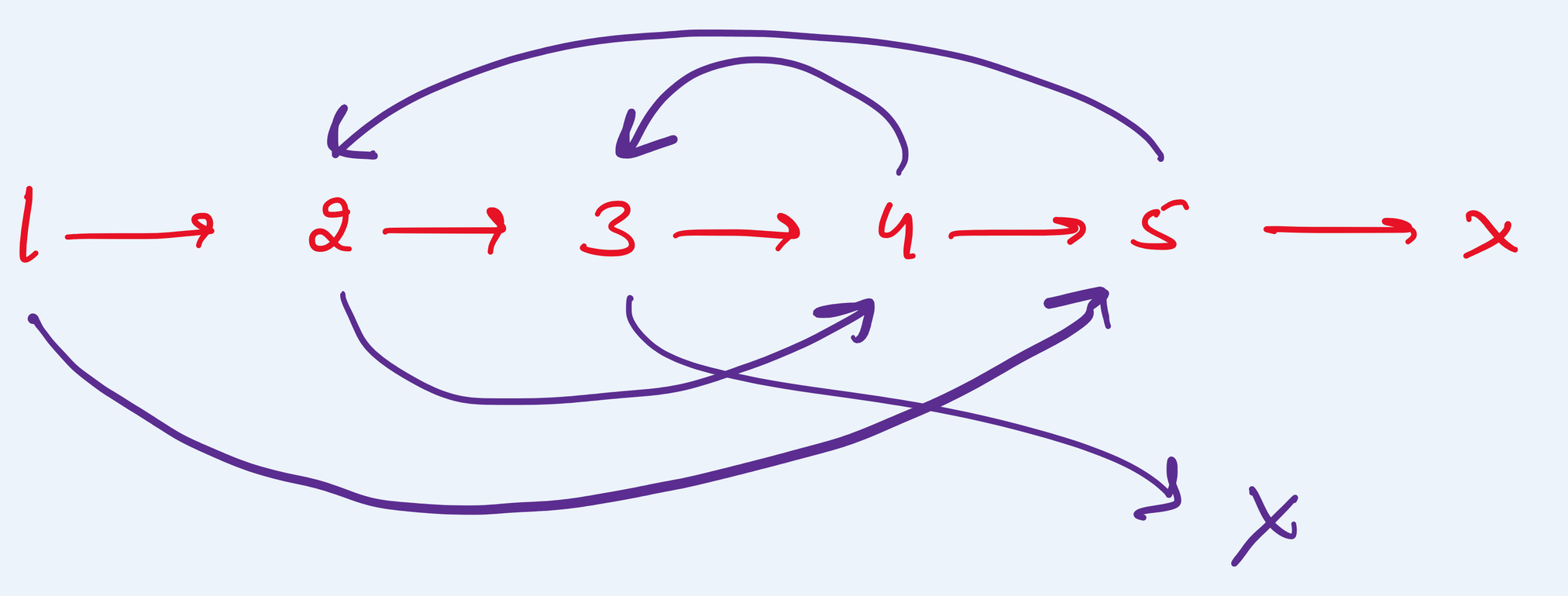
Q Given a linked list, reorder / folding linked list

$L_0 \to L_1 \to L_2 \to L_3 - - - - - \; L_{n-1} \to L_n$

$L_0 \to L_n \to L_2 \to L_{n-2} \leftarrow - - - -$

(1) Eg: 1 → 2 → 3 → 4 → x
ans → 1 → 4 → 2 → 3 → x

(2) Eg: 1 → 2 → 3 → 4 → 5 → x
ans → 1 → 5 → 2 → 4 → 3 → x



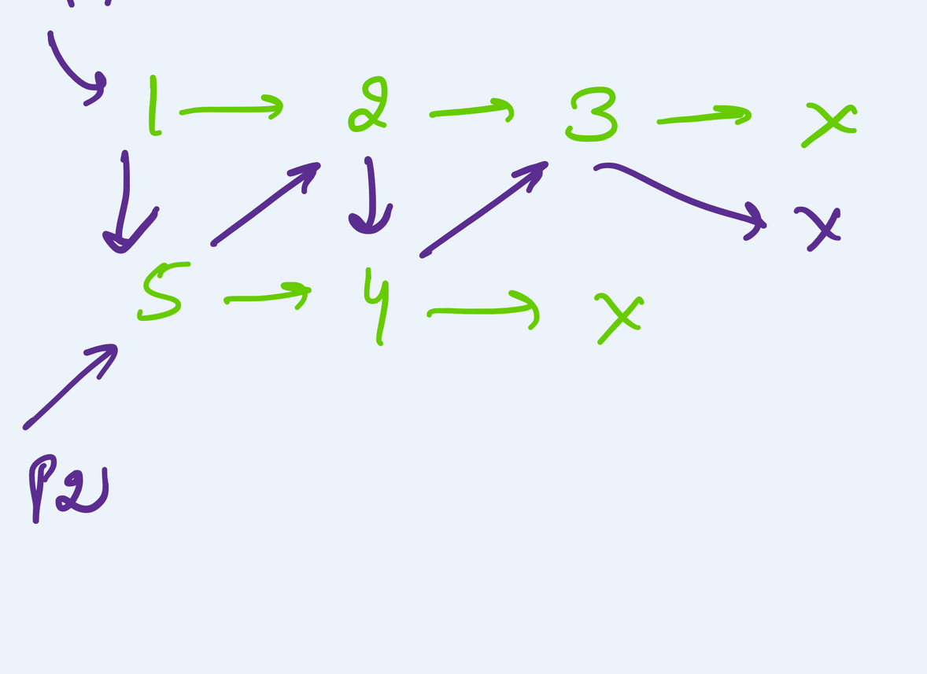
Approach:

1 → 2 → 3 → 4 → 5 → x

1. Find the mid of LL ✓
mid = (3)

2. Reverse the second half of LL ✓
1 → 2 → 3 → x
4 → 5 → x
5 → 4 → x

3. Start reordering both linked list one by one. → merge linked list



Q Detect a cycle in linked list

(a) 3 → 2 → 0 → 4 (4 points back to 2) — true

(b) 1 → 2 → 3 → 4 → x — false

(c) 1 → 2 → 3 → 4 → 5
8 ← 7 ← 6 (5 → 6, 8 → 3) — true