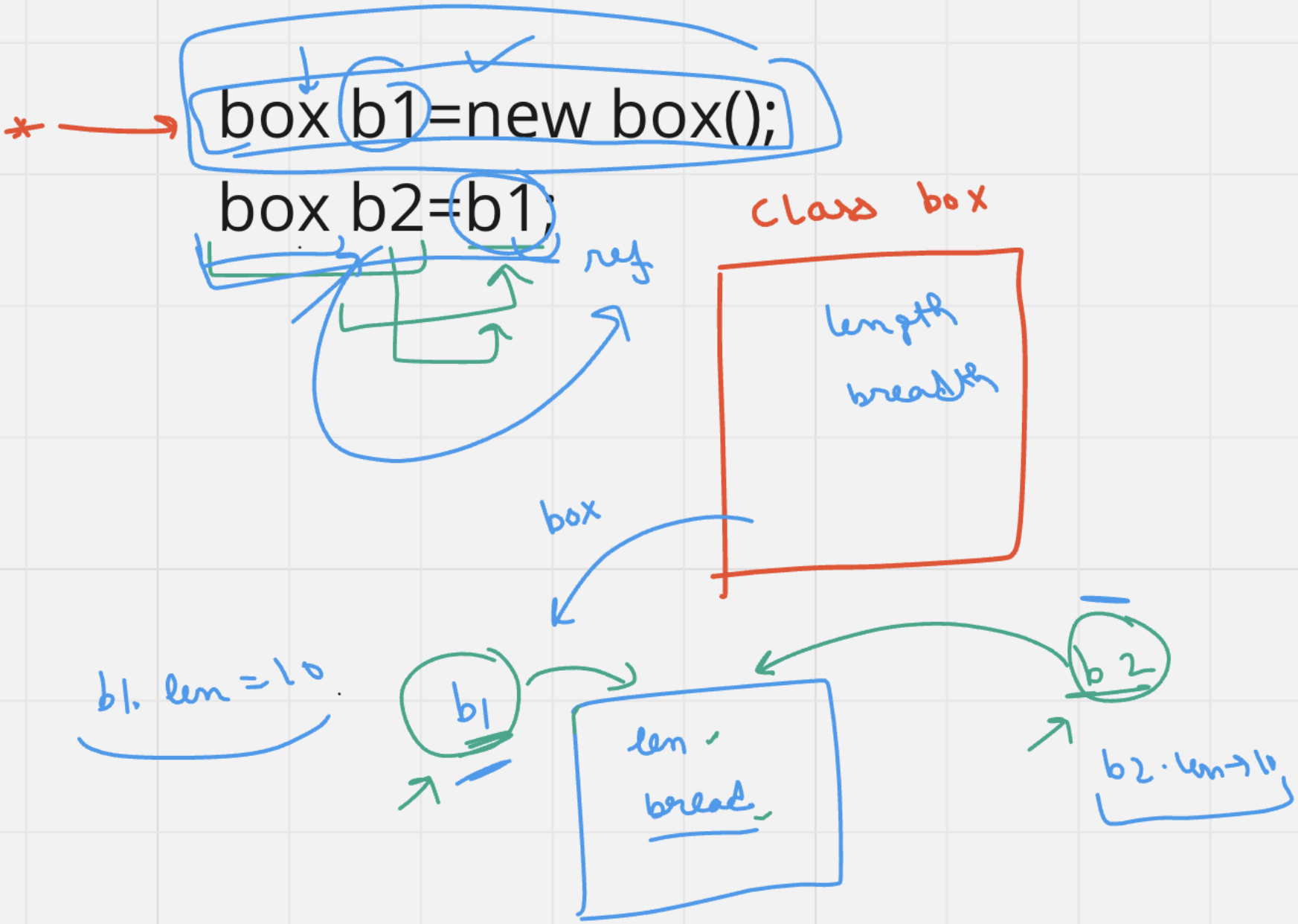

box b1=new box();
box b2=b1;
ref
Class box
length
breadth
box
b1. len = 10
b1
b2
len
bread
b2. len → 10

Linked List

↳ Node's

↓

int data

↳

ref of next

Node ← address

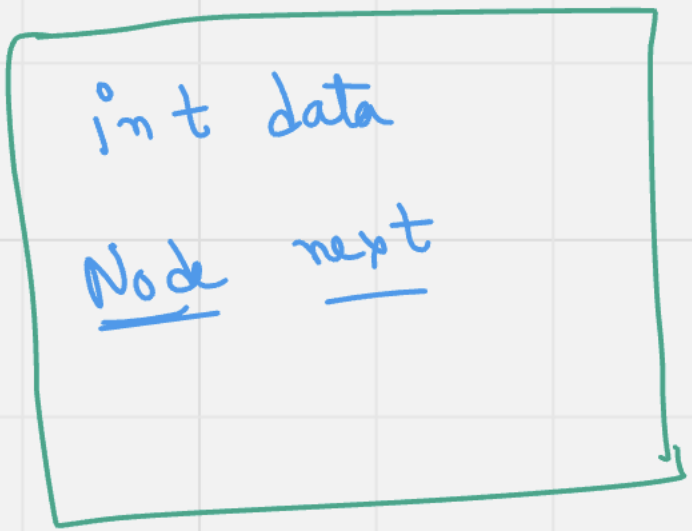
Pair ←

int

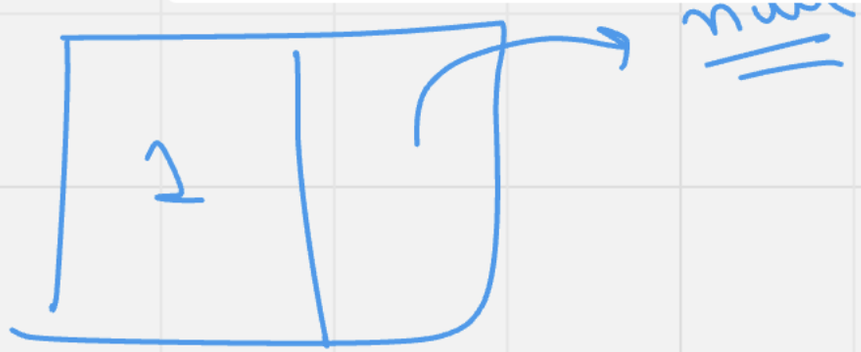
string

Pair p

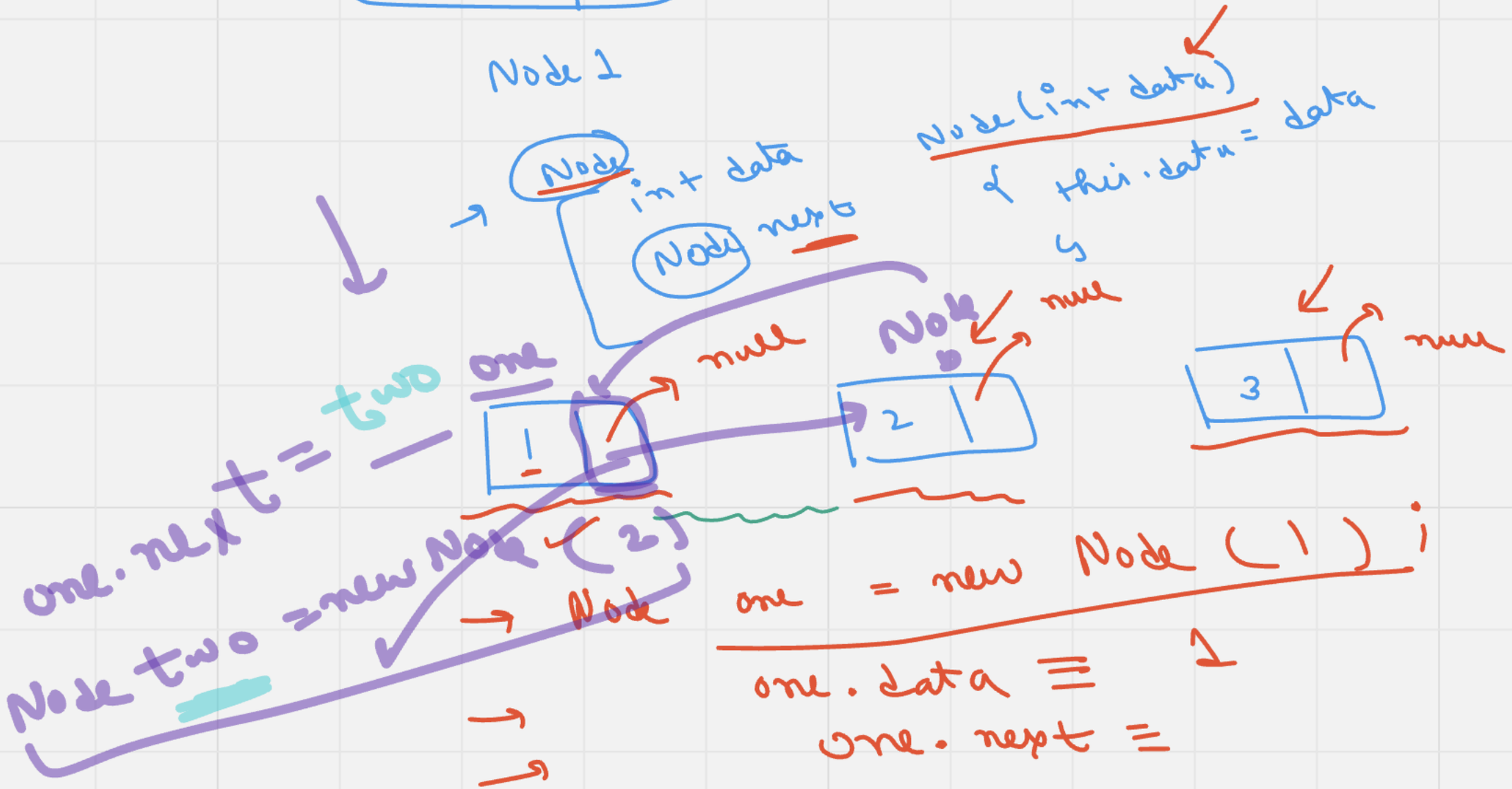
class Node ←



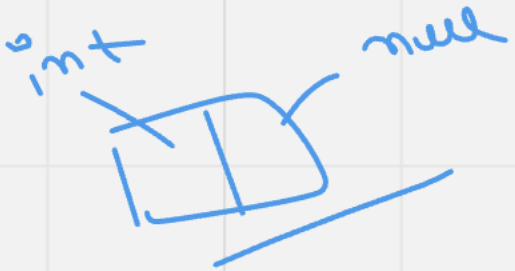
Node


1

Node 1

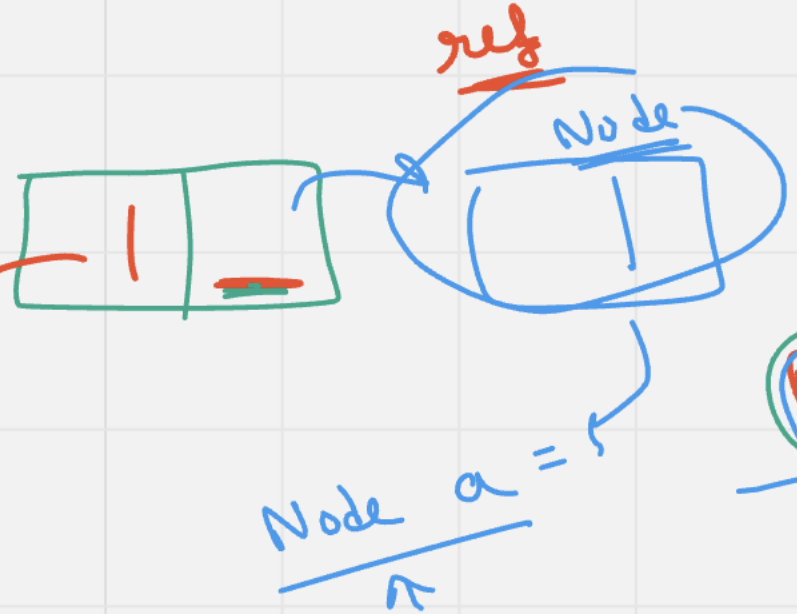

Node
int data
Node next
Node(int data)
{ this.data = data
}
one.next = two
Node two = new Node (2)
one
null
1
2
3
null
null
Node
one = new Node (1);
one.data ≡ 1
one.next ≡

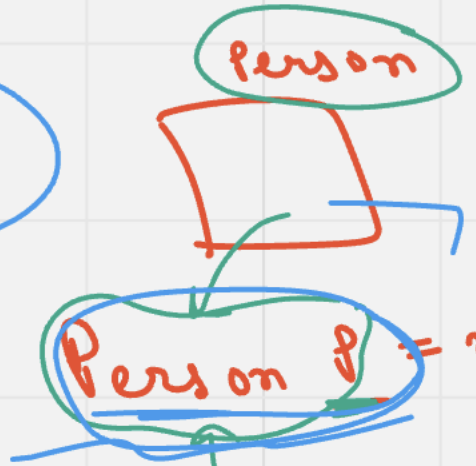
```
class Node {
    int data;
    Node next;

    public Node(int data) {
        this.data = data;
    }
}
```



null

new Node use ref





Node two = new Node (2)

one.next = two

one.data

1

one.next.data

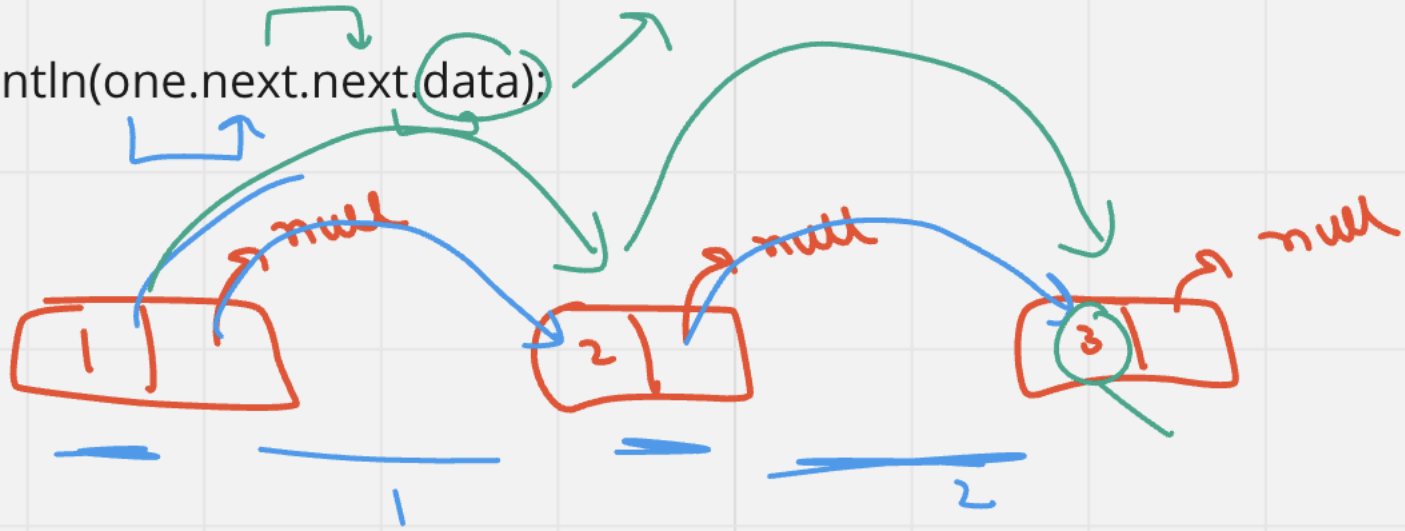
null Pointer exception

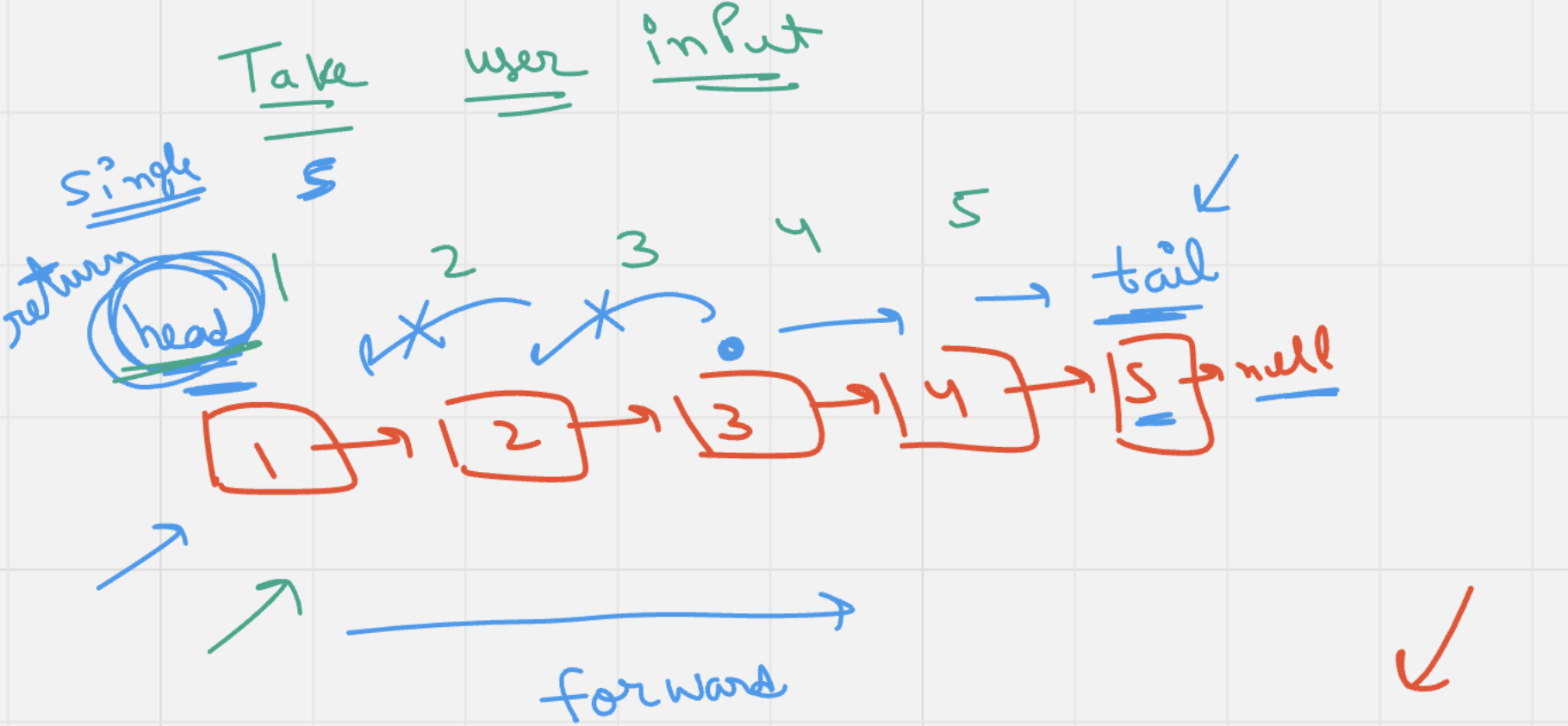
(2) ← null.data

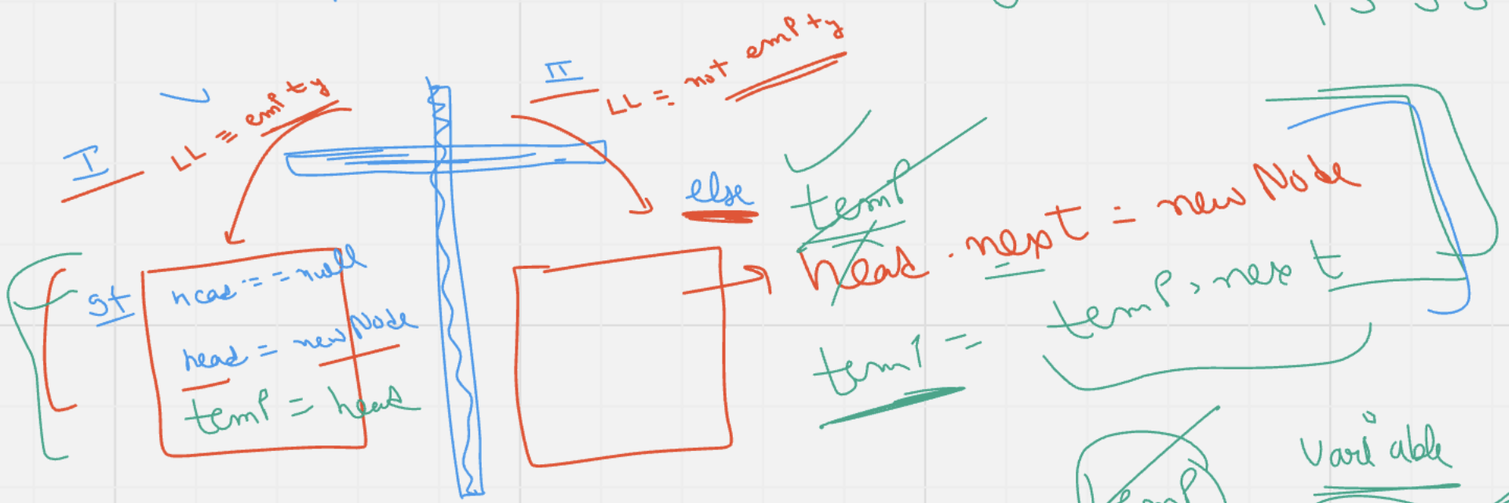
one.next.next.data

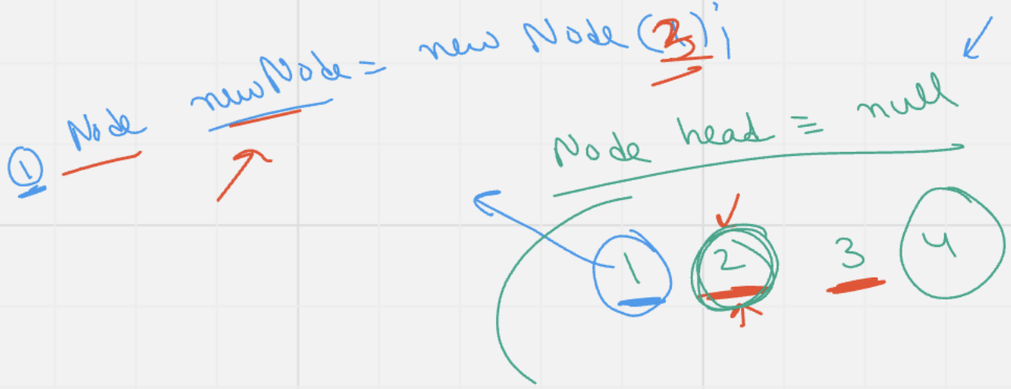
3

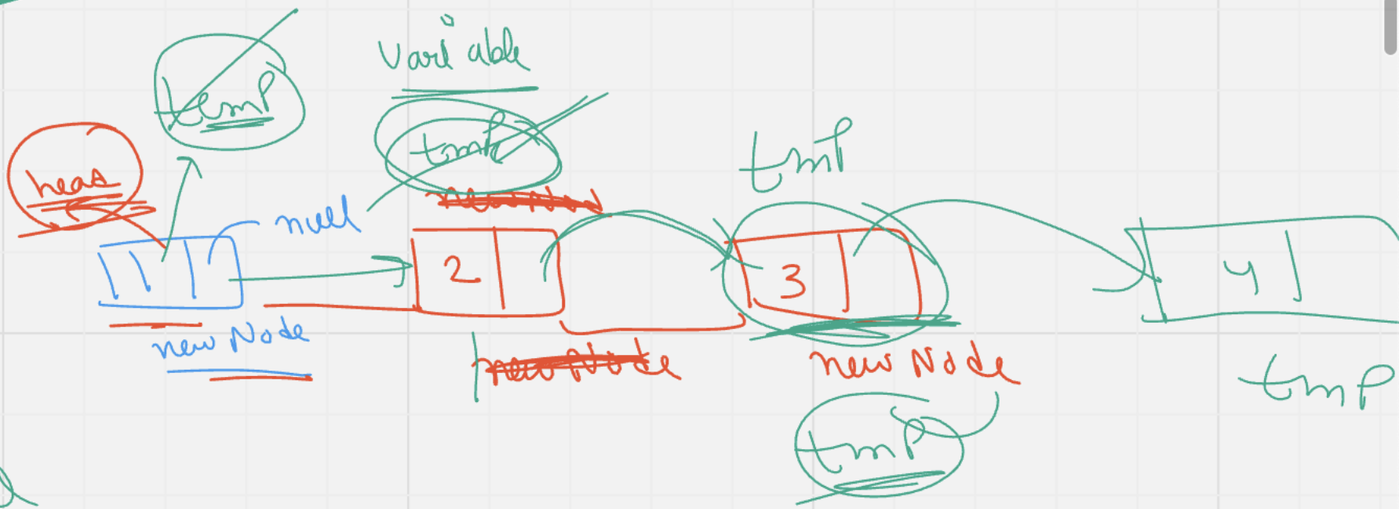
```
Node one = new Node(1);
Node two = new Node(2);
Node three = new Node(3);

one.next = two;
two.next = three;

System.out.println(one.next.next.data);
```




Take user input
5
Single
return
head
1
2
3
4
5
tail
1
2
3
4
5
null
forward


I
LL = empty
II
LL = not empty
else
if
head == null
head = newNode
temp = head
temp
head.next = newNode
temp1 = temp.next


① Node newNode = new Node(2);
Node head = null
1
2
3
4
Node temp = null


temp
Variable
temp
head
null
2
3
4
new Node
temp
new Node
temp

Node head = null

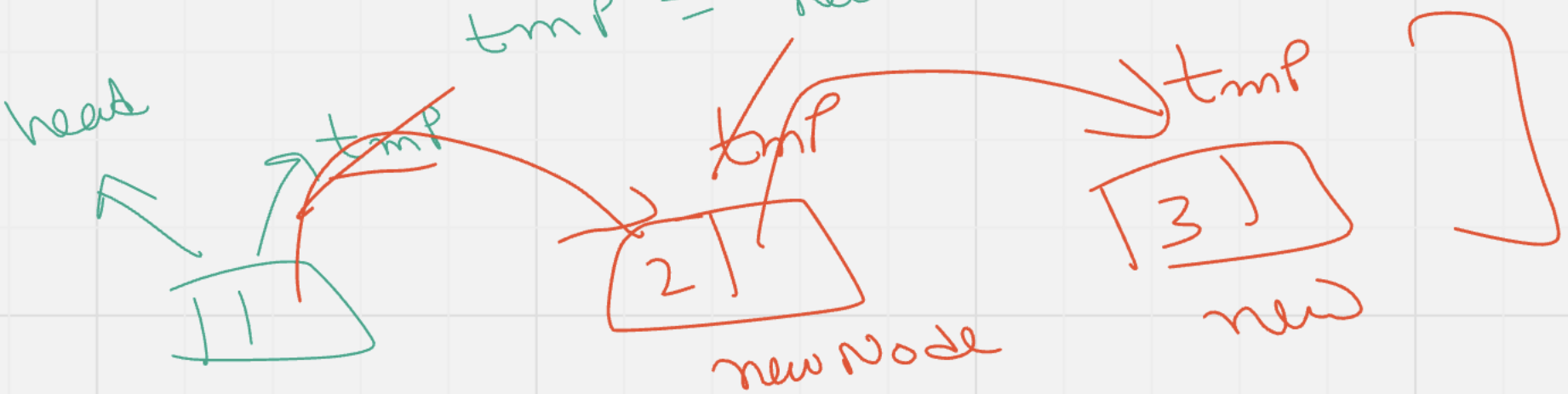
Node tmp = null

head = newNode

tmp = newNod

tmp.next = nn

temp = temp.ng



Print LL

stop

while (head != null)

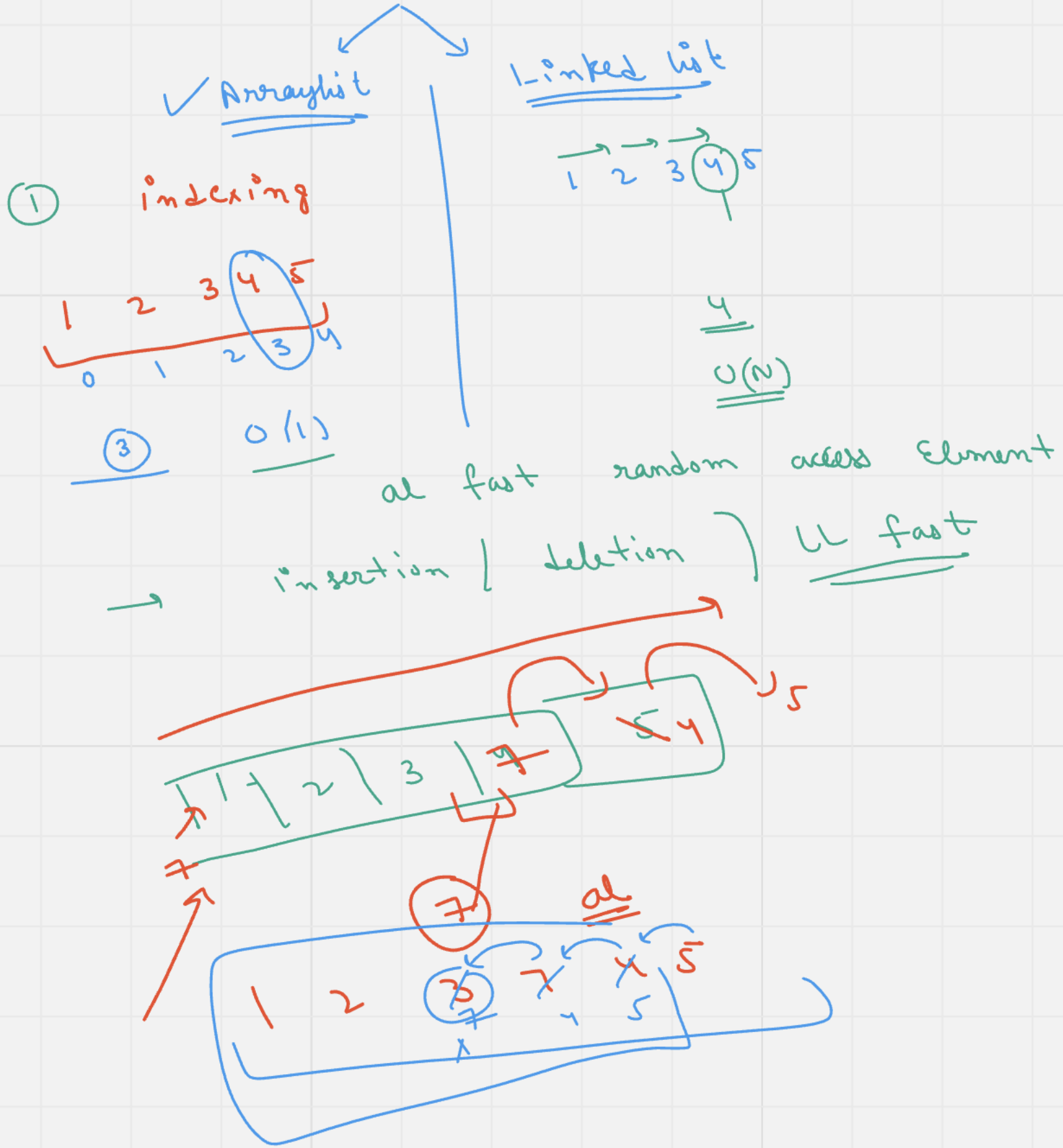
{

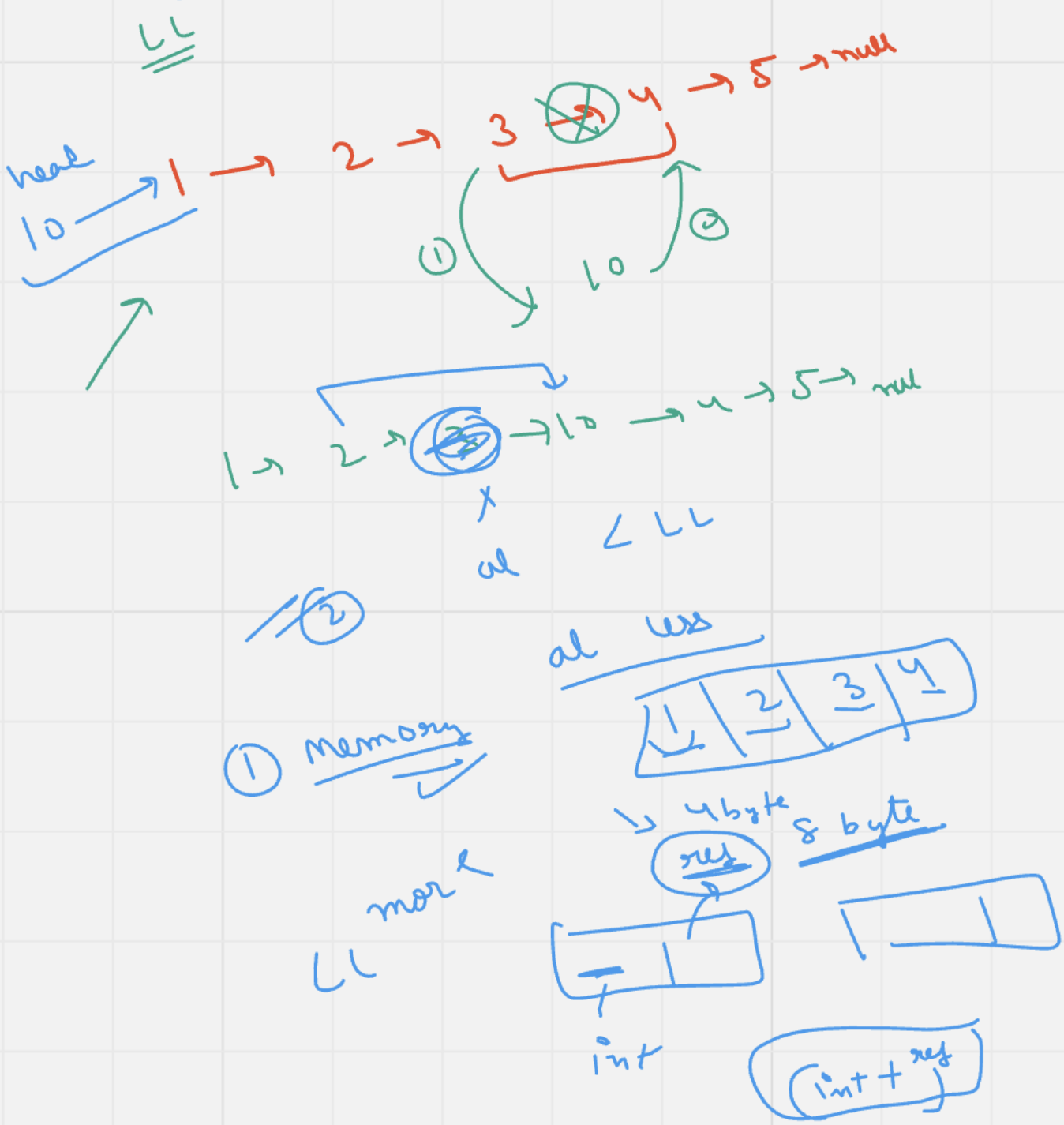
sop (head.data);

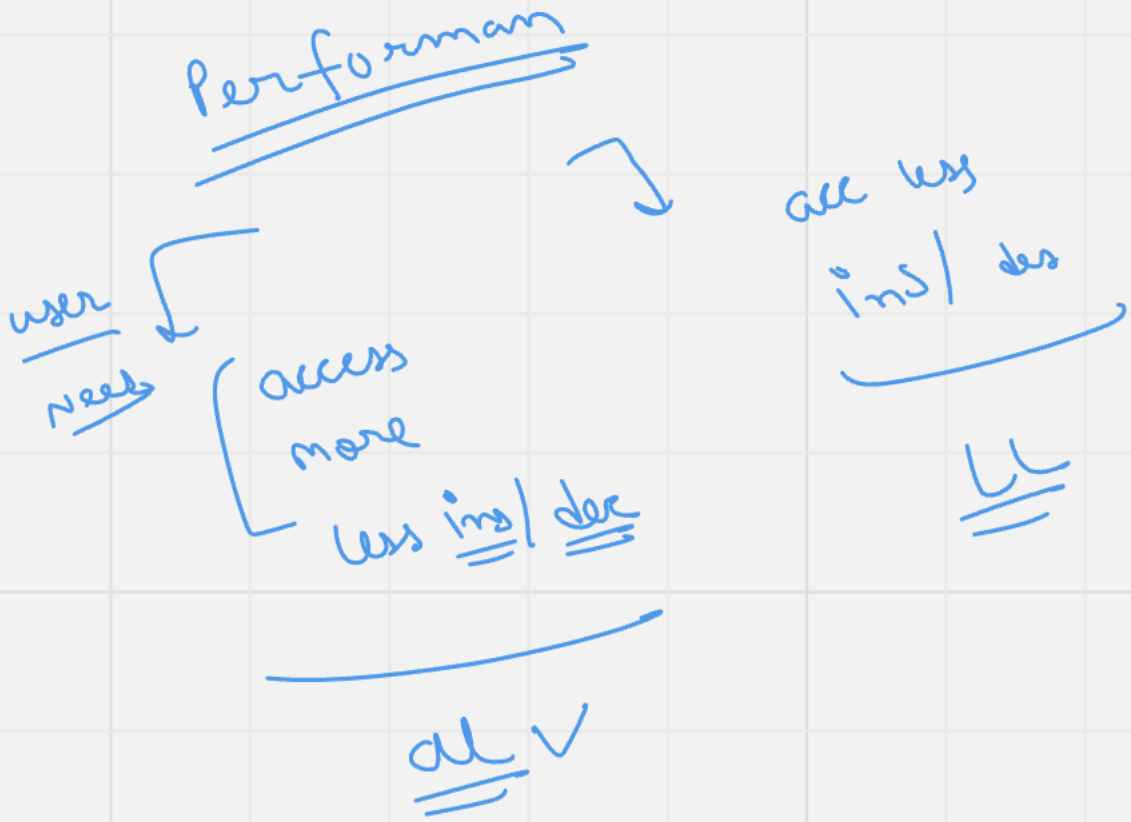
head = head.next

}

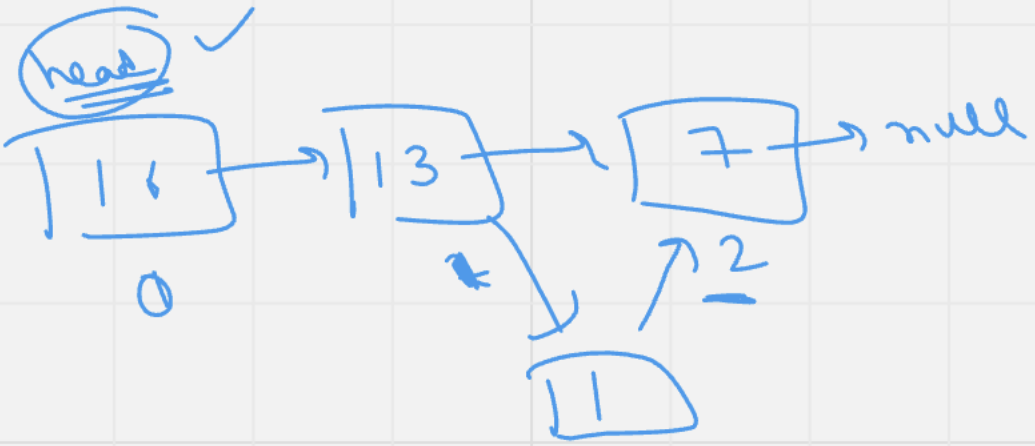
select


Arraylist
Linked list
1 indexing
1 2 3 4 5
0 1 2 3 4
3
O(1)
1 2 3 4 5
4
O(N)
al fast random access Element
insertion | deletion | LL fast
5
5 4
1 2 3
7
7
al
1 2 3 7 X 5
4 5


LL
head
10
1
2
3
4
5
null
1
2
10
4
5
null
X
LL
2
memory
address
1
2
3
4
4 byte
8 byte
ref
int
int + ref
LL
more

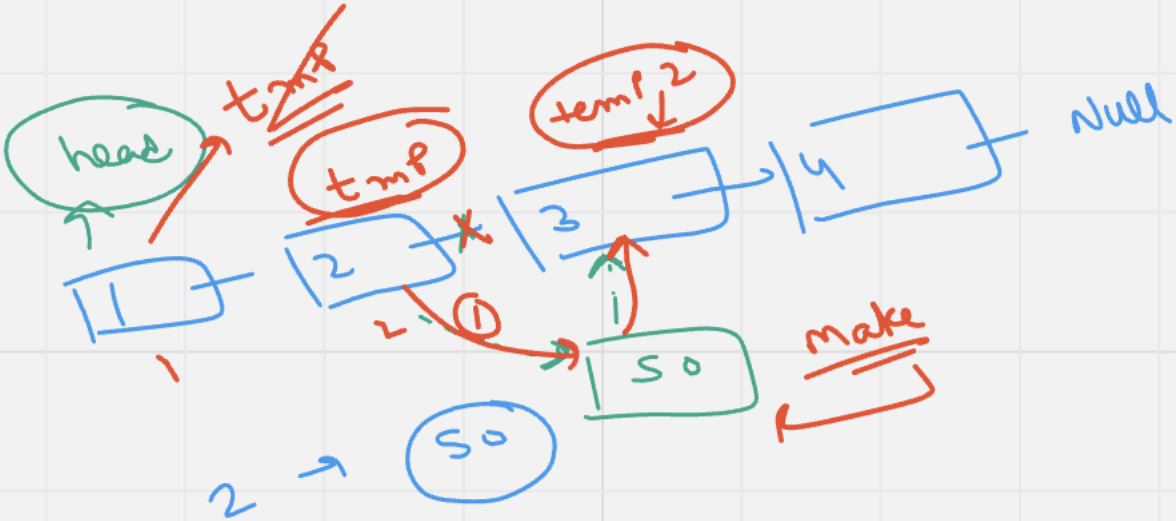

Performan
user needs
access more less ins/ dec
acc less ins/ dec
LL
dl V

3
16 13 7
1 2



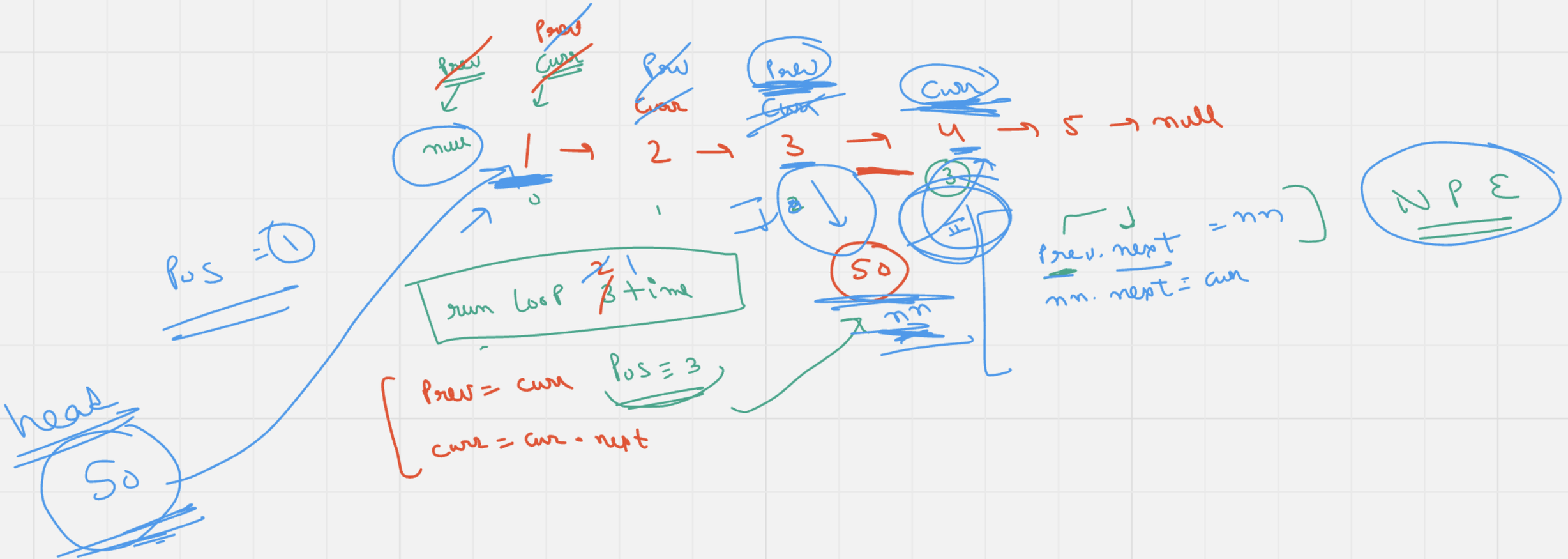
length
n = 3
Pos = 2
Val = 1

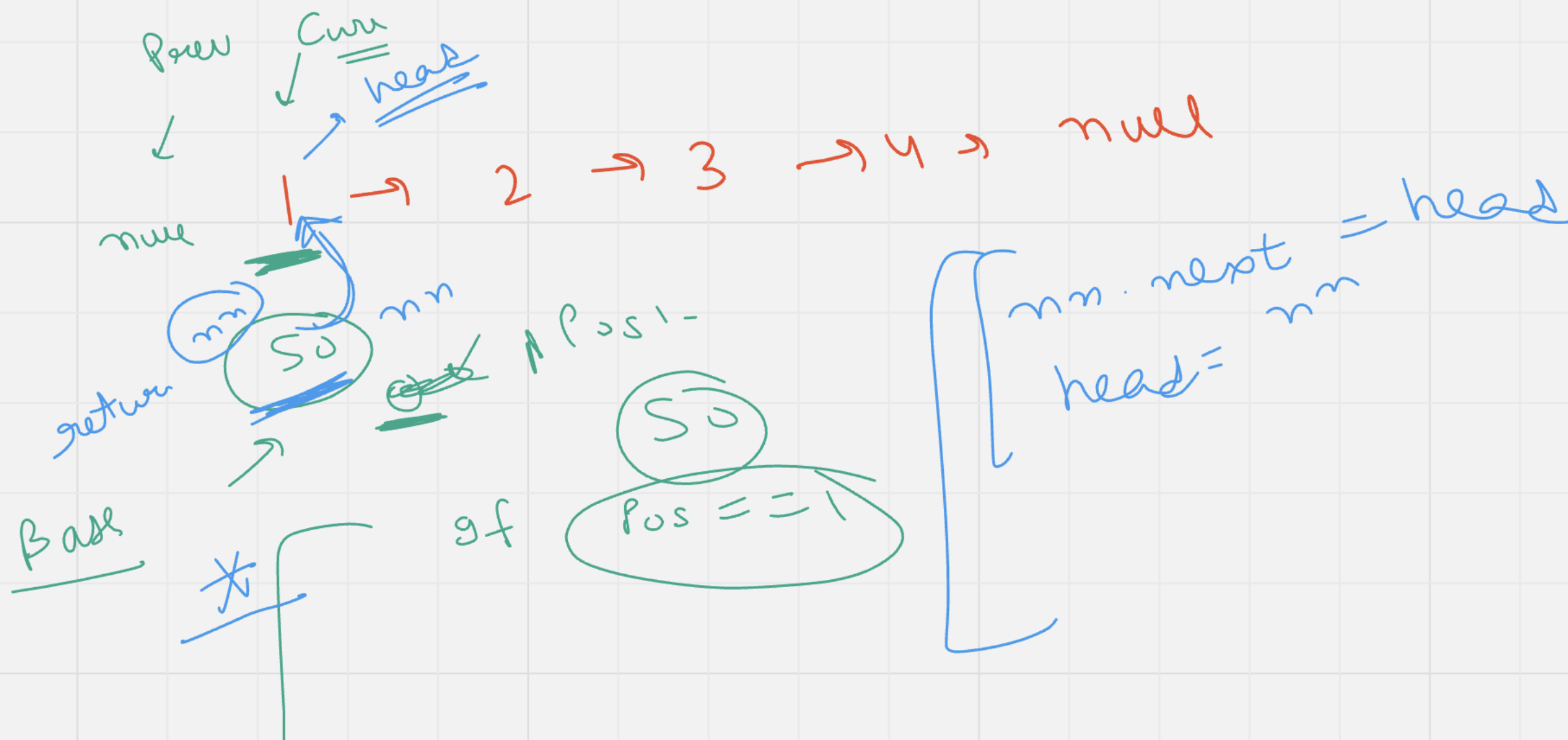
16 → 13 → 1 → 7

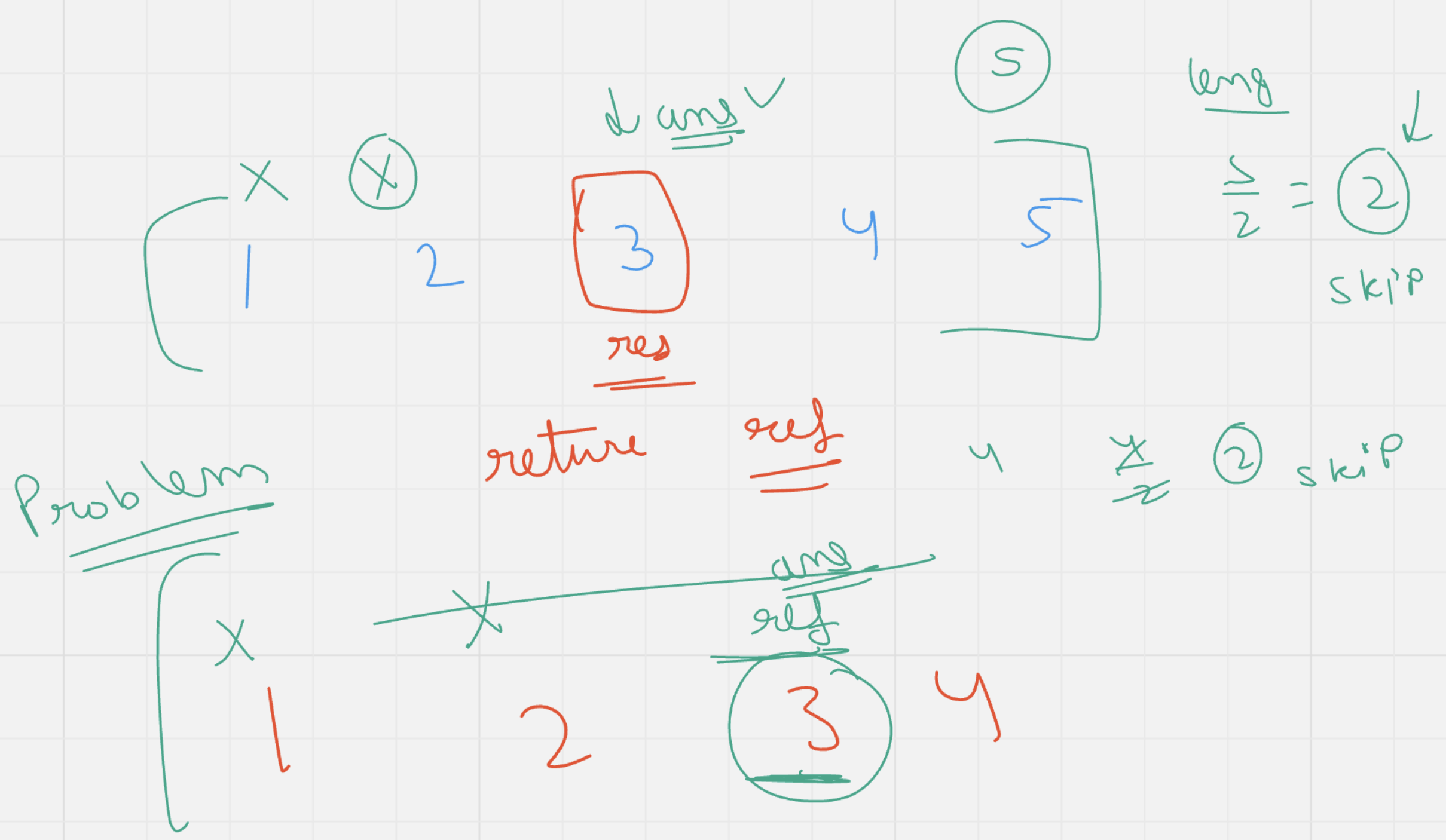


2 →

```
tmp.next = 50
50.next = temp2
return head
```


head
50
pos = 1
run loop 2 1 3 time
prev = curr
curr = curr.next
pos = 3
null
1
2
3
4
5
null
50
nn
prev.next = nn
nn.next = curr
NPE


Prev
Curr
head
1 → 2 → 3 → 4 → null
null
mm
nn
SO
return
Base
pos--
SO
pos == 1
gf
nn.next = head
nn
head =


X
X
1
2
3
res
4
5
5
leng
2
skip
return
ref
4
2
skip
Problem
X
1
2
3
4
ans
ref

10 20 30 40 50

len → 5

skip = len/2 = 2

```
while(skip-- > 0) {
    head = head.next;
}
```

Two time

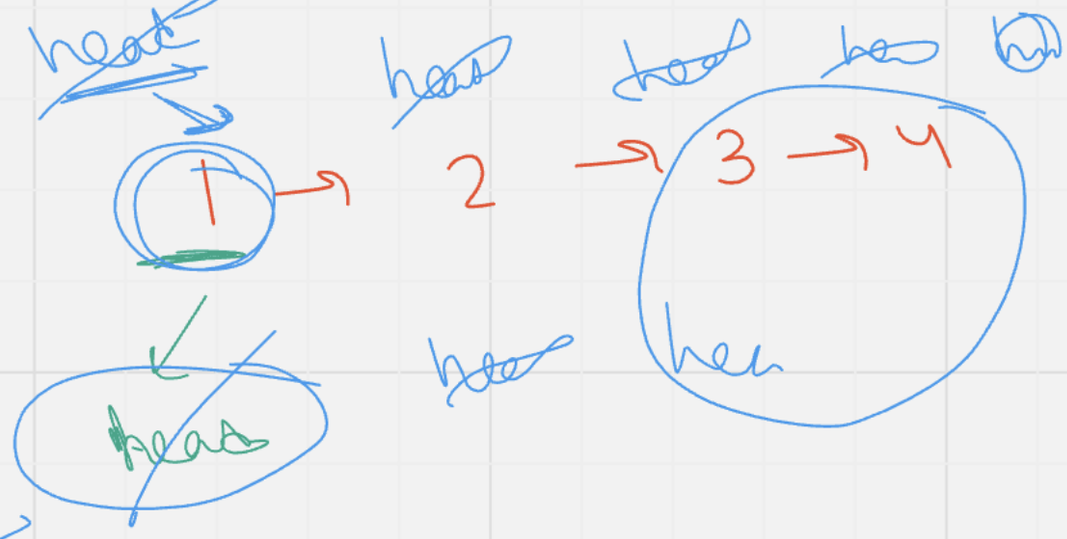
2 > 0 True

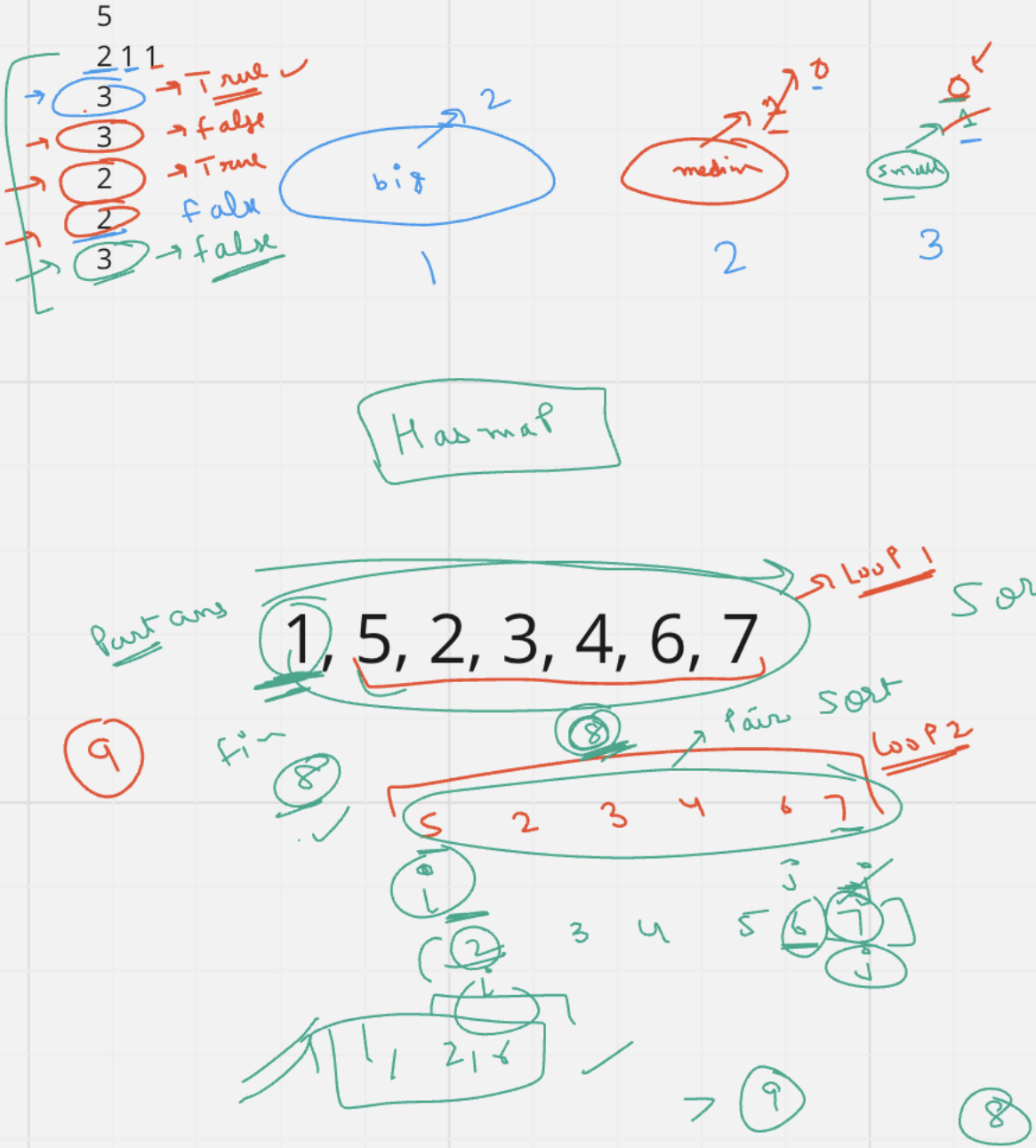
1 > 0 True

0 > 0 False

while (skip > 0)
{
    skip--;
}

```
static int LenOfLL(Node head) {

    int cnt = 0;
    while(head != null) {
        head = head.next;
        cnt++;
    }
    return cnt;

}

static Node midpointOfLinkedList(Node head)
{
    int len = LenOfLL(head);

    int skip = len / 2;

    while(skip-- > 0) {
        head = head.next;
    }

    return head;

}
```




5
2 1 1
3
3
2
2
3
True
false
True
false
false
big
2
1
median
2
small
3
Hashmap
Part ans
1, 5, 2, 3, 4, 6, 7
Loop 1
Sort
fair sort
Loop 2
fi~
5 2 3 4 6 7
i
j
3 4 5 6 7
1 2 4
9
8