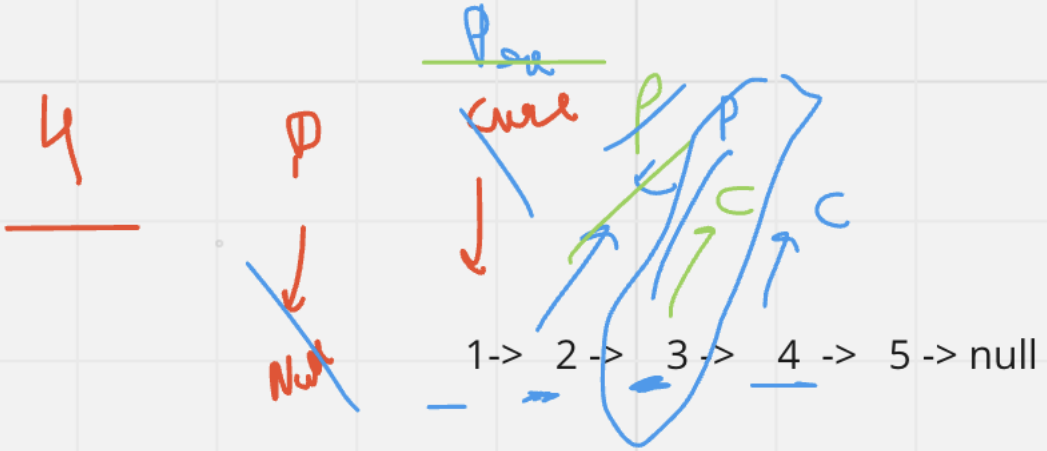
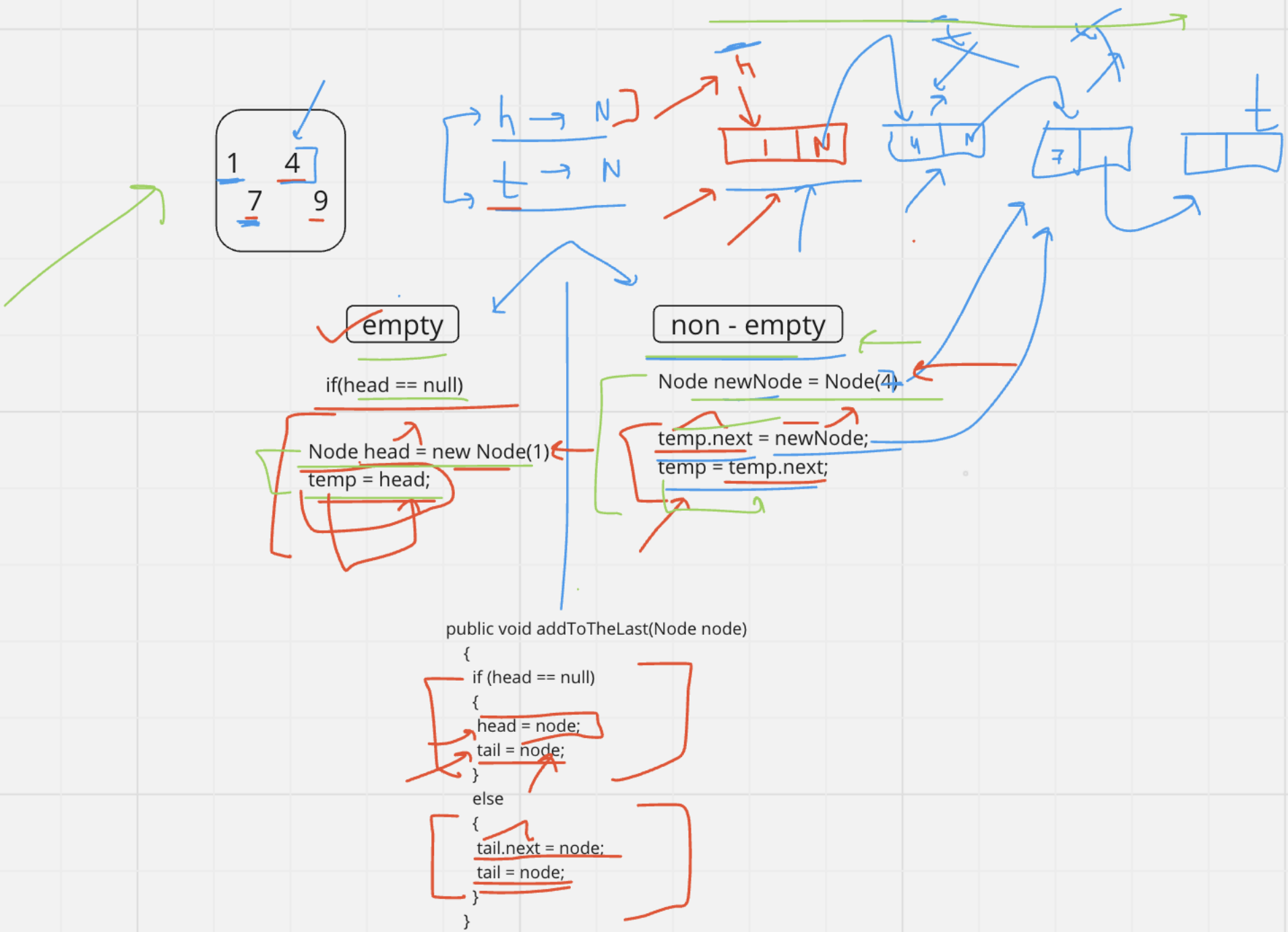
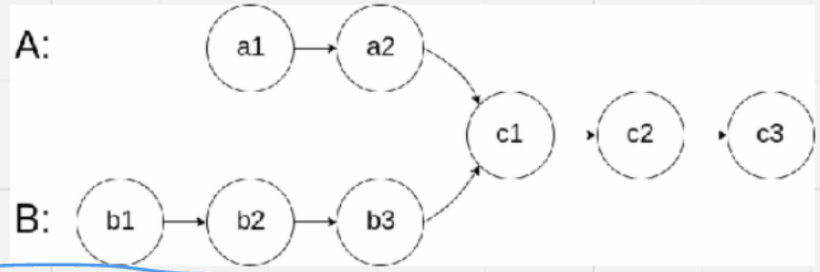


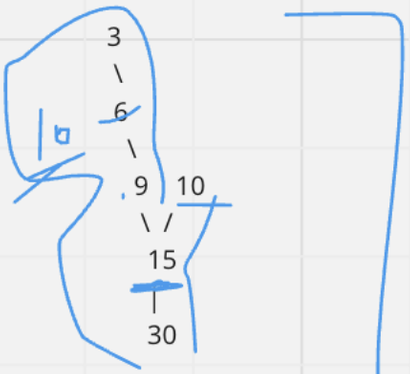

```
Node curr = head;
Node prev = null;

while(curr != null) {

    if(curr.data == val) {
        prev = curr
        break
    }
    prev = curr;
    curr = curr->next;

}
```


1
4
7
9
h → N
t → N
empty
non - empty
if(head == null)
Node head = new Node(1)
temp = head;
Node newNode = Node(4)
temp.next = newNode;
temp = temp.next;
public void addToTheLast(Node node)
{
if (head == null)
{
head = node;
tail = node;
}
else
{
tail.next = node;
tail = node;
}
}



L1: 3 - 6 - 9 - 15 - 30

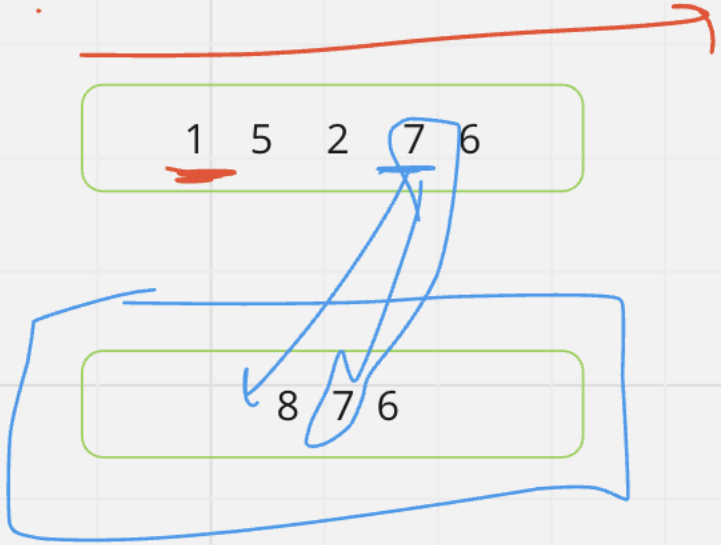
L2: 10 - 15 - 30

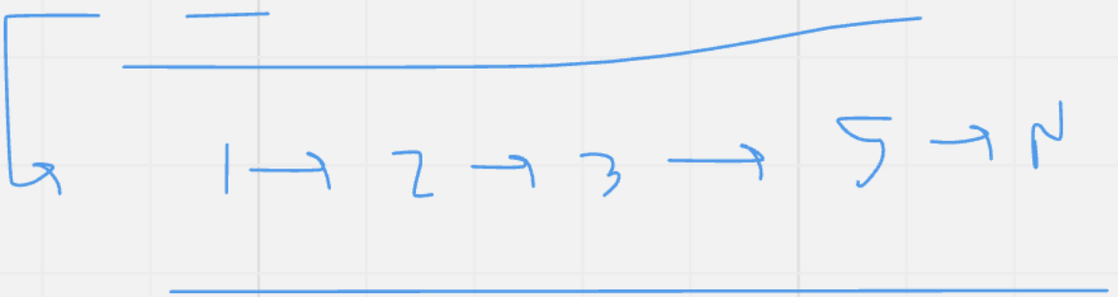


1 5 2 7 6

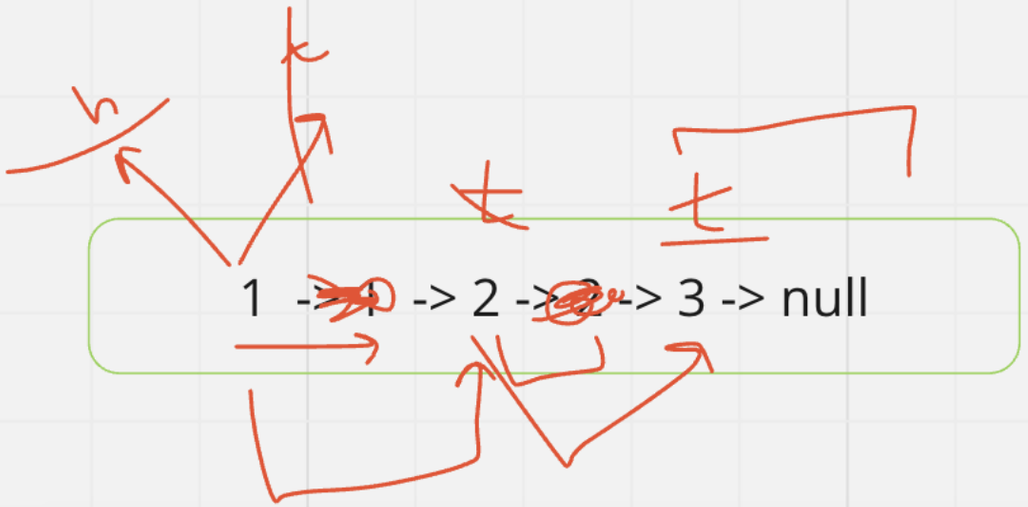
8 7 6

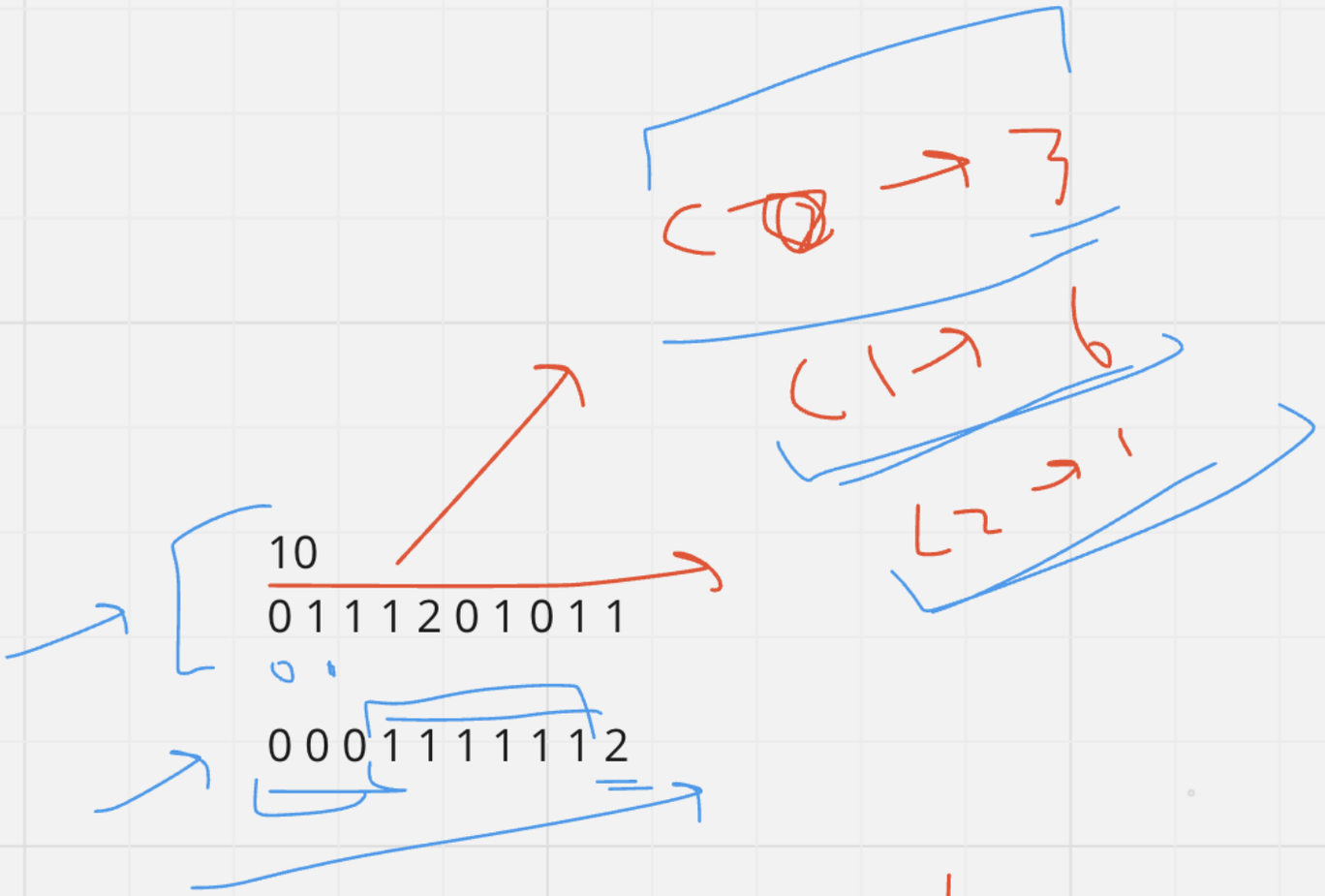
```
while(headA != null) {

    Node temp = headB;

    while(temp != null) {

      if(headA == temp) {
        return headA;
      }else {
          temp = temp.next;
      }
    }

    headA = headA.next;


}
```

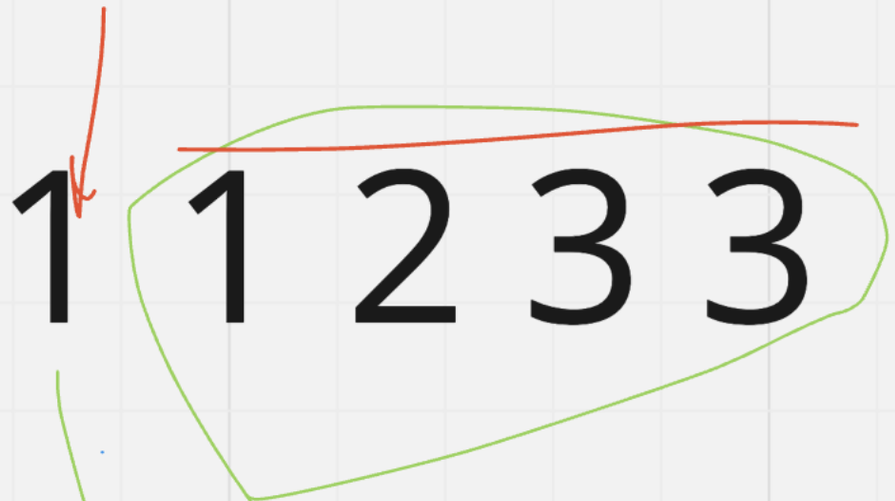


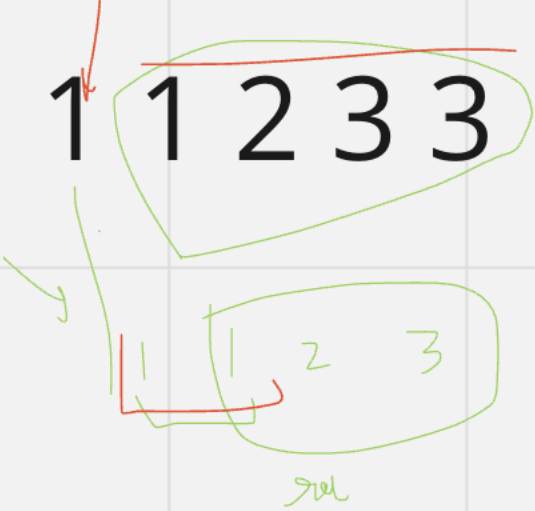


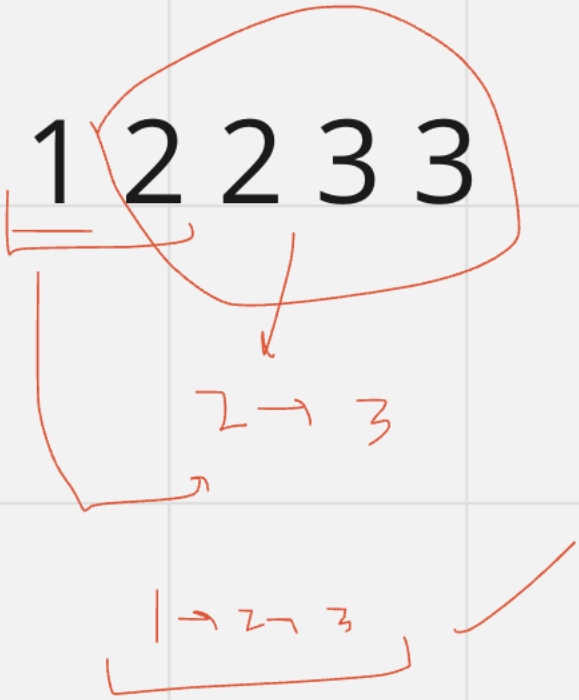
1->1->2->2->2->2->3->5->null

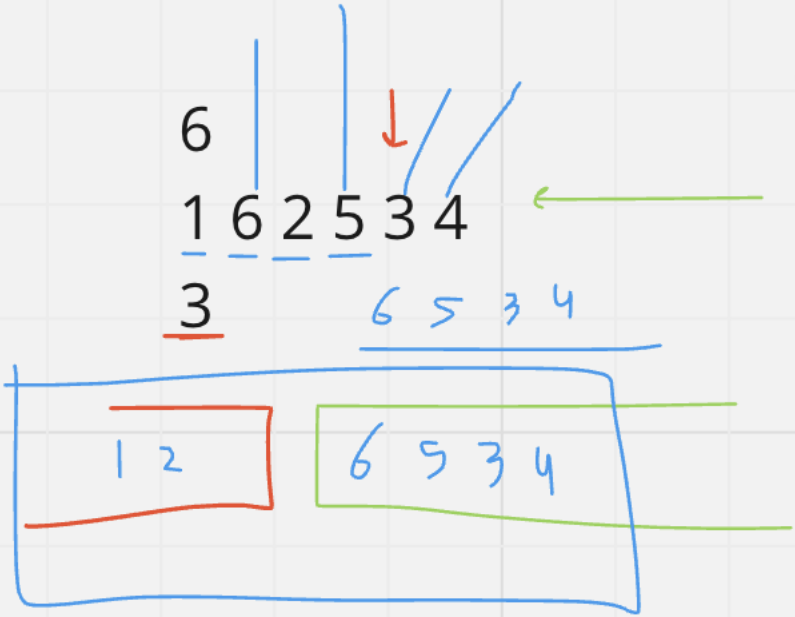


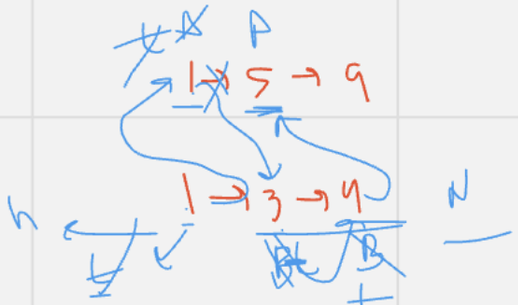

10
0 1 1 1 2 0 1 0 1 1
0 0 0 1 1 1 1 1 1 2

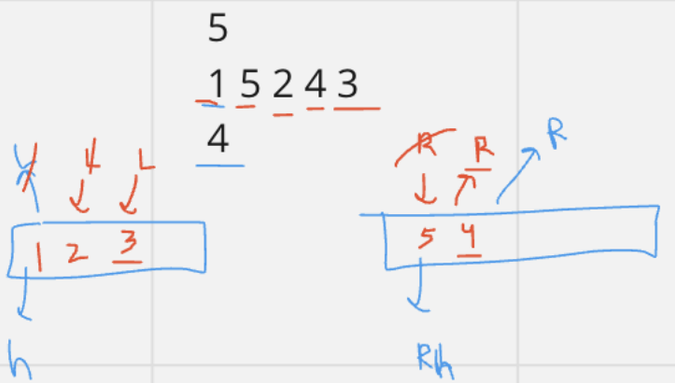

1 1 2 3 3

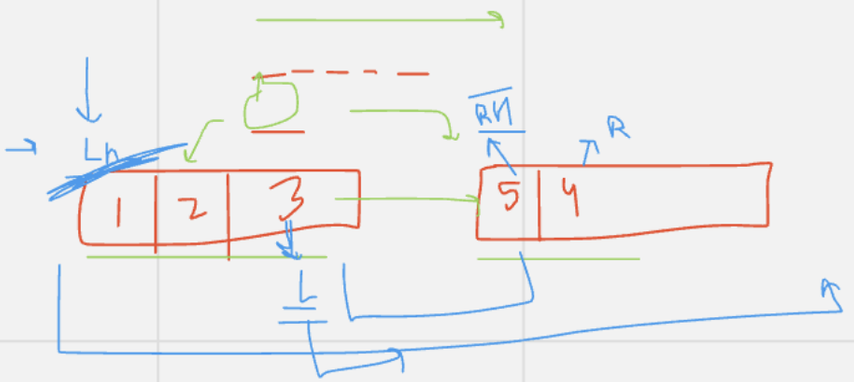

1 1 2 3 3
1 1 2 3


1 2 2 3 3
2 → 3
1 → 2 → 3


6
1 6 2 5 3 4
3
6 5 3 4
1 2
6 5 3 4







```
while(A != null && B != null)
{

if(A.data < B.data) {

temp.next = A;
temp = temp.next;

}
```

3

1->5 ->9

3

1->3->4

```
if ( A.data < B.data)
{
    head = A;
    temp = head;
}
else {
    head = B;
    temp = head;
}
```

```
}
```