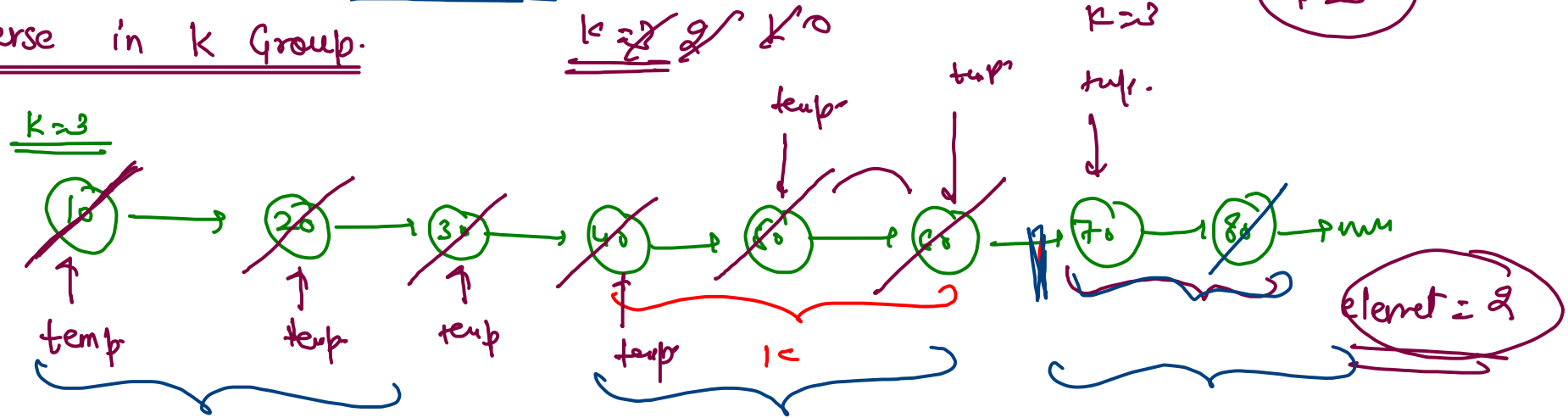
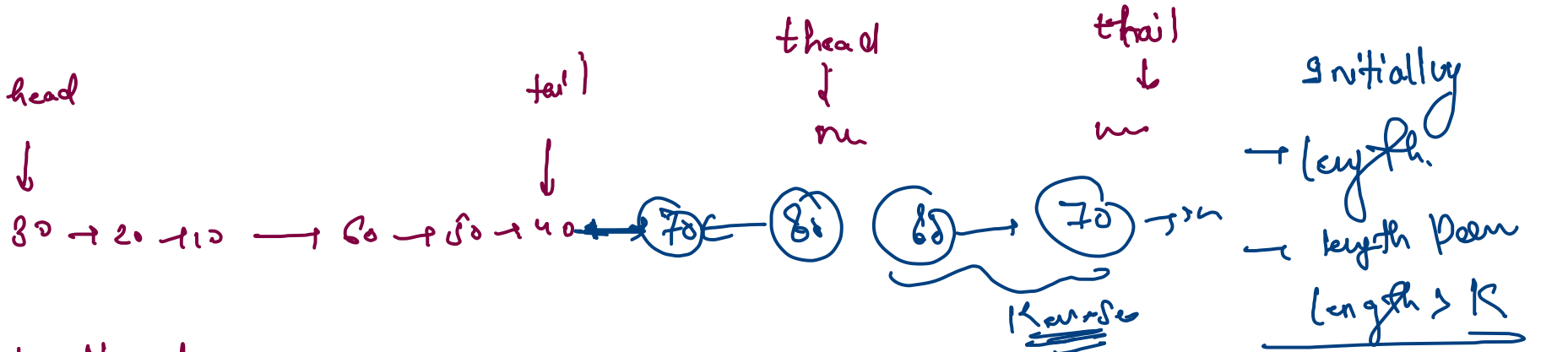
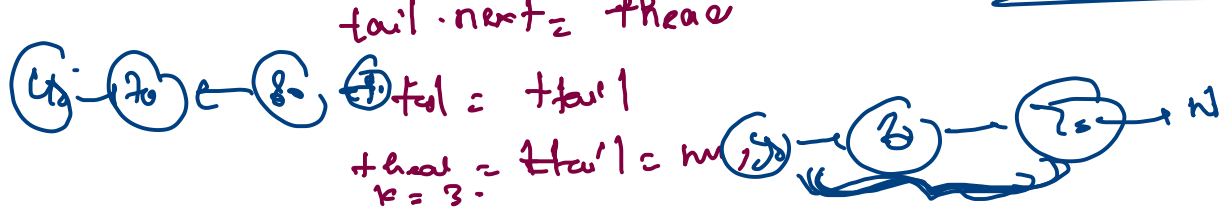
Space Not allowed

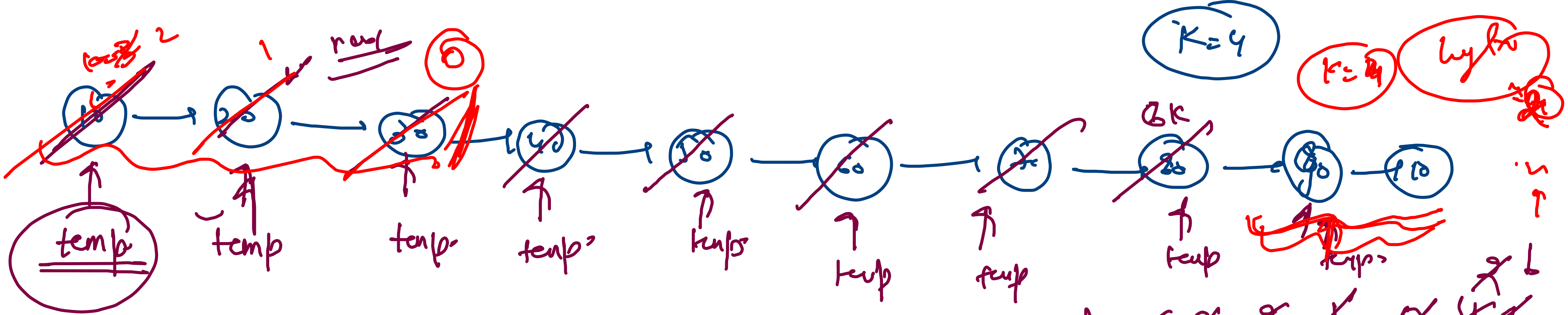
Reverse in k Group.





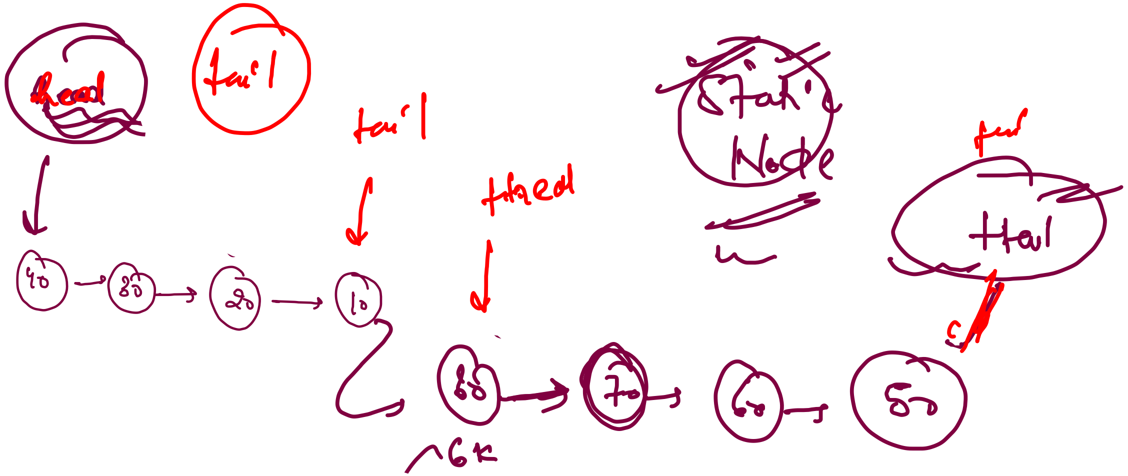

```
thead = thead;
tail = ttail;
```





k=4 3 2 1 0 4 3

length: 10 9 8 7 6 5 4



① Find length.

(Preserve current of temp)

make add first call to temp

temp = preserve Node

k--;

length--

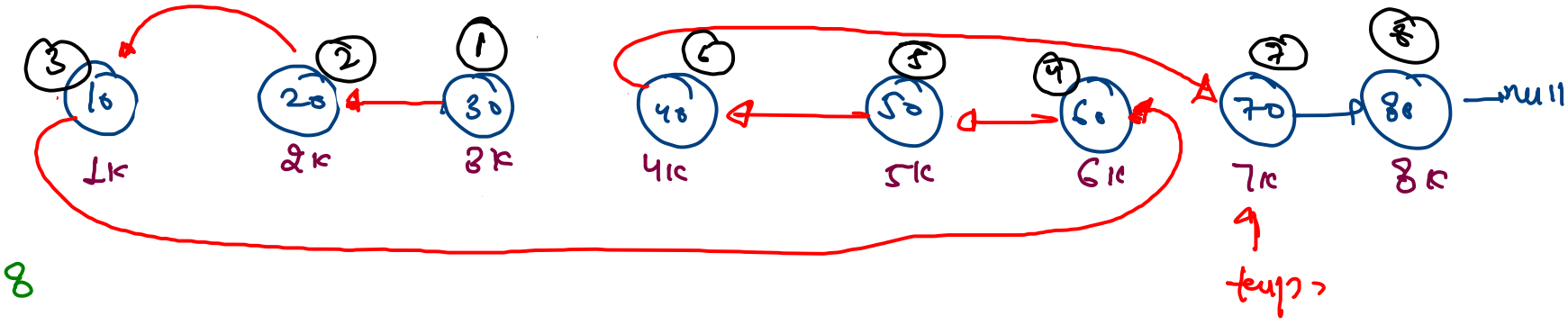
k==0 → head, tail set

tail.next = fhead;

tail = ftail;

fhead = ftail = null;

k=0



K = 2

length : 8

```java
public static ListNode reverseInKGroup(ListNode node, int K) {
    if(node == null || node.next == null || K == 0) return node;

    int len = length(node);

    thead = ttail = null;

    ListNode head = null;
    ListNode tail = null;
    ListNode temp = node;
    int k = K;
    while(len >= k) {
        ListNode next = temp.next;
        addFirst(temp);
        temp = next;
        k--;
        len--;

        if(k == 0) {
            if(head == null) {
                head = thead;
                tail = ttail;
            } else {
                tail.next = thead;
                tail = ttail;
            }
            thead = ttail = null;
            k = K;
        }
    }

    tail.next = temp;
    return head;
}
```

length = 2

k = 2

thead = null

ttail = null

head = 3K

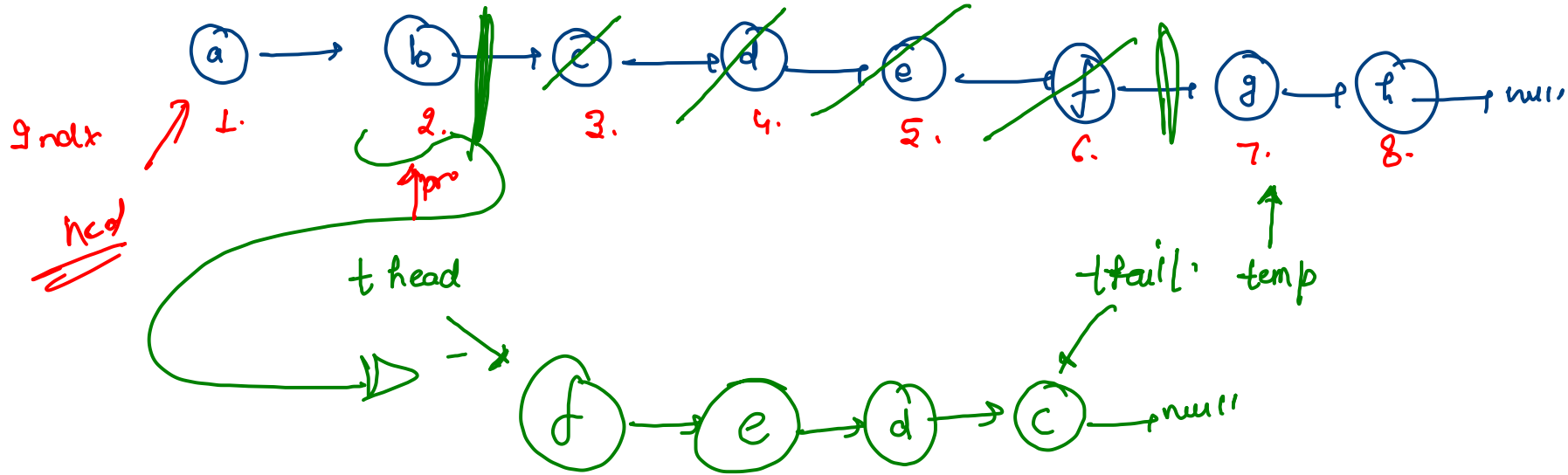
tail = 4K

return head = 3K

```java
public static ListNode thead = null;
public static ListNode ttail = null;

public static void addFirst(ListNode node) {
    if(thead == null) {
        thead = ttail = node;
    } else {
        node.next = thead;
        thead = node;
    }
}
```

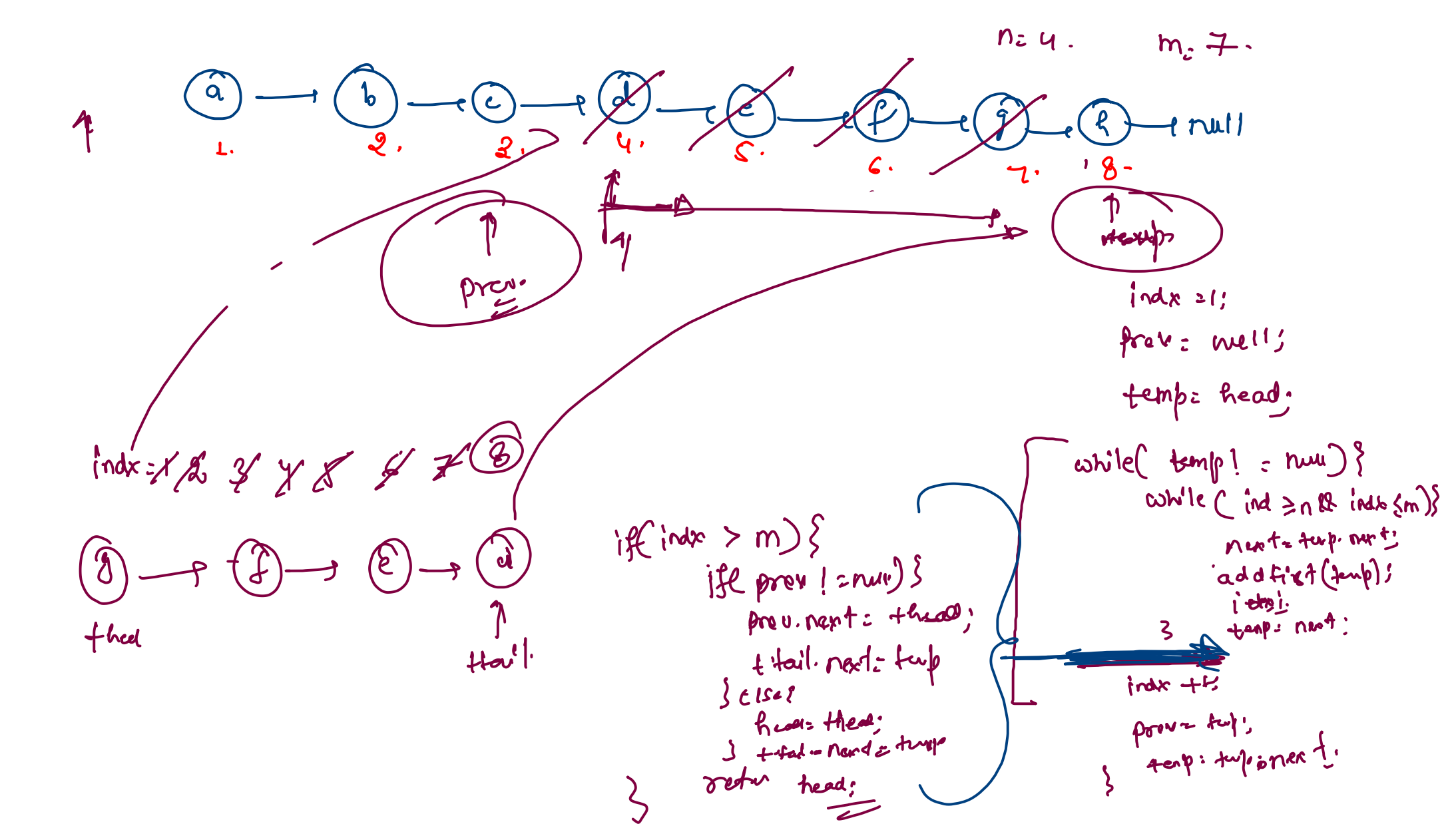
Reverse In Range:

n = 3, m = 6



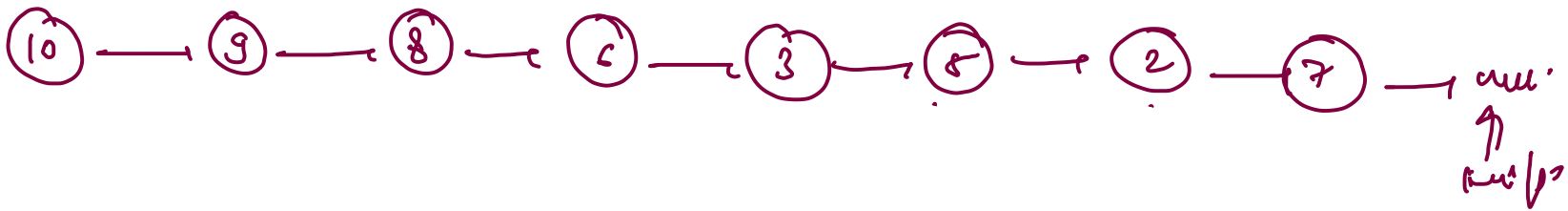

```
tail.next = temp;
prev.next = thead;
return head;
```

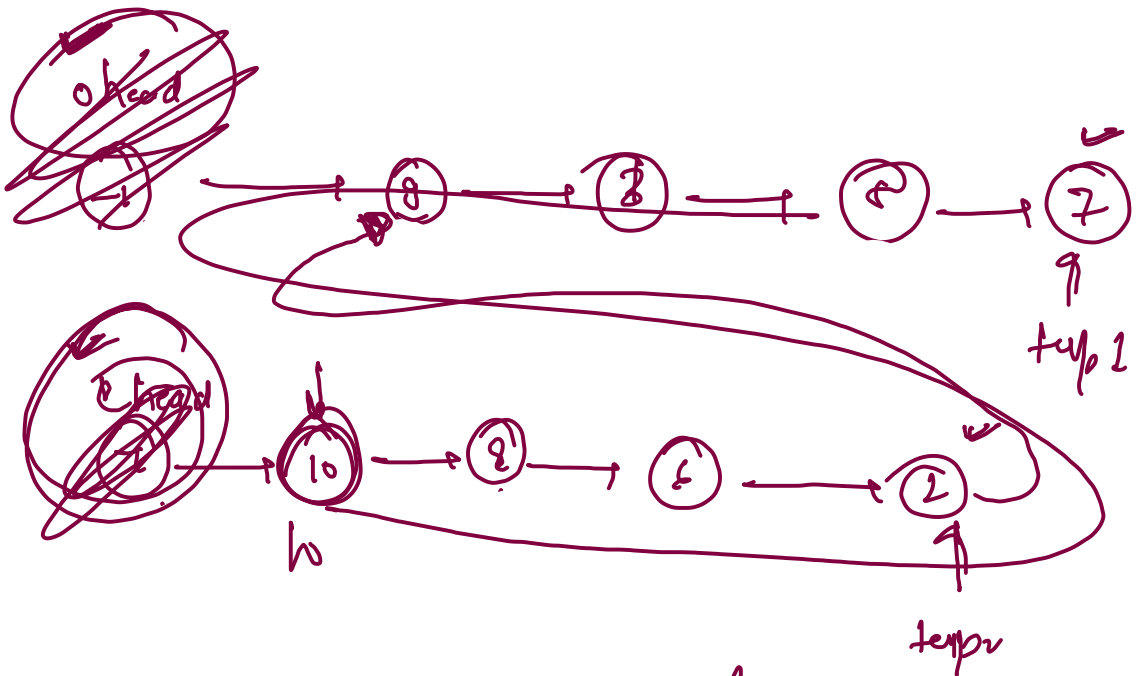
Indx = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ 7


n = 4.
m = 7.
a
b
c
d
e
f
g
h
null
1.
2.
3.
4.
5.
6.
7.
8.
prev.
temp
indx = 1;
prev = null;
temp = head;
while(temp != null){
while (indx >= n && indx < m){
next = temp.next;
addFirst(temp);
i++;
temp = next;
indx ++;
prev = temp;
temp = temp.next;
}
if(indx > m){
if prev != null){
prev.next = thead;
t.tail.next = temp
}else{
head = thead;
} t.tail = next = temp
return head;
}
indx = 1 2 3 4 5 6 7 8
g
f
e
d
thead
tail.

Segregate odd Even Linked List :

i/p

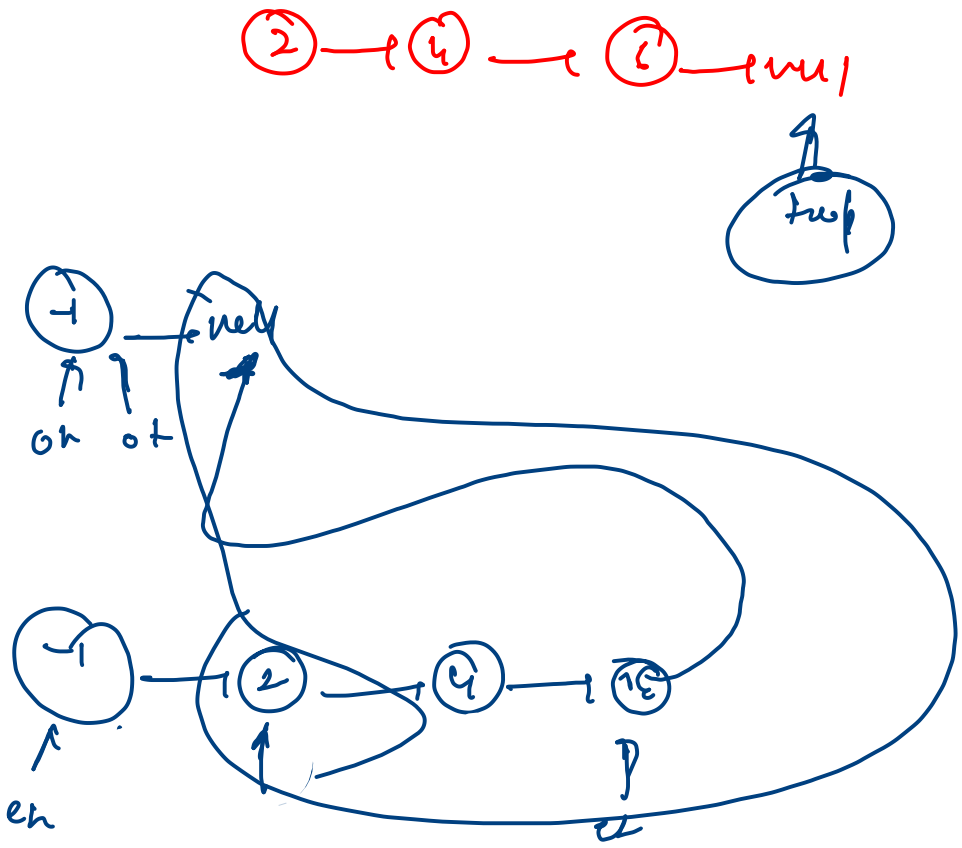




temp2.next = ohead.next

return ehead.next

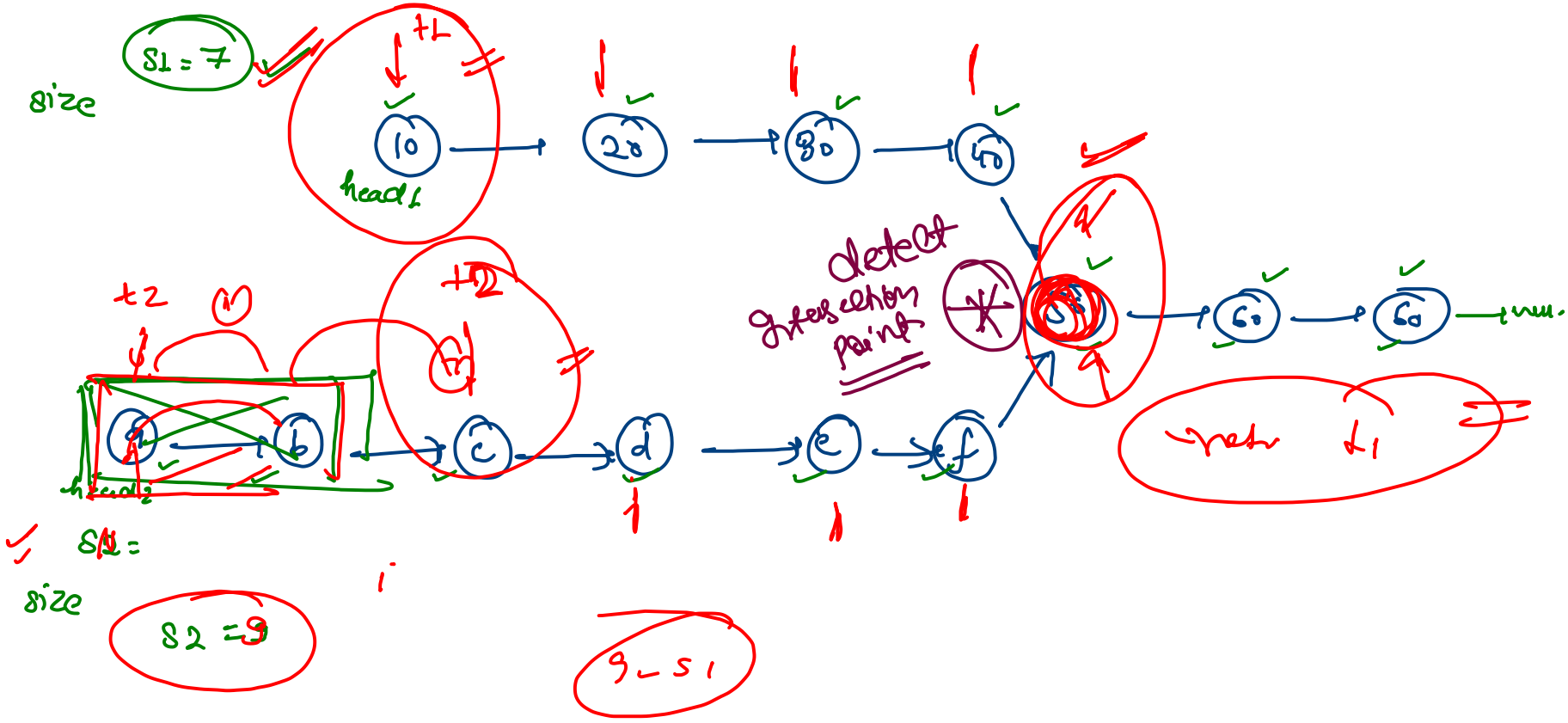
```
if(temp.val % 2 == 0){

  temp2.next = temp;
  temp = temp.next;
  temp2 = temp2.next
} else {

  temp1.next = temp;

  temp = temp.next;
  temp1 = temp1.next;
}
```




```java
public static ListNode segregateEvenOdd(ListNode head) {
    // dummy nodes
    ListNode ohead = new ListNode(-1);
    ListNode ehead = new ListNode(-1);

    // make tail of odd list and even list for iteration
    ListNode otail = ohead, etail = ehead;

    ListNode temp = head;

    while(temp != null) {
        if(temp.val % 2 == 0) {
            // even
            etail.next = temp;
            etail = temp;
        } else {
            // odd
            otail.next = temp;
            otail = temp;
        }
        temp = temp.next;
    }

    etail.next = ohead.next;
    otail.next = null;
    return ehead.next;
}
```

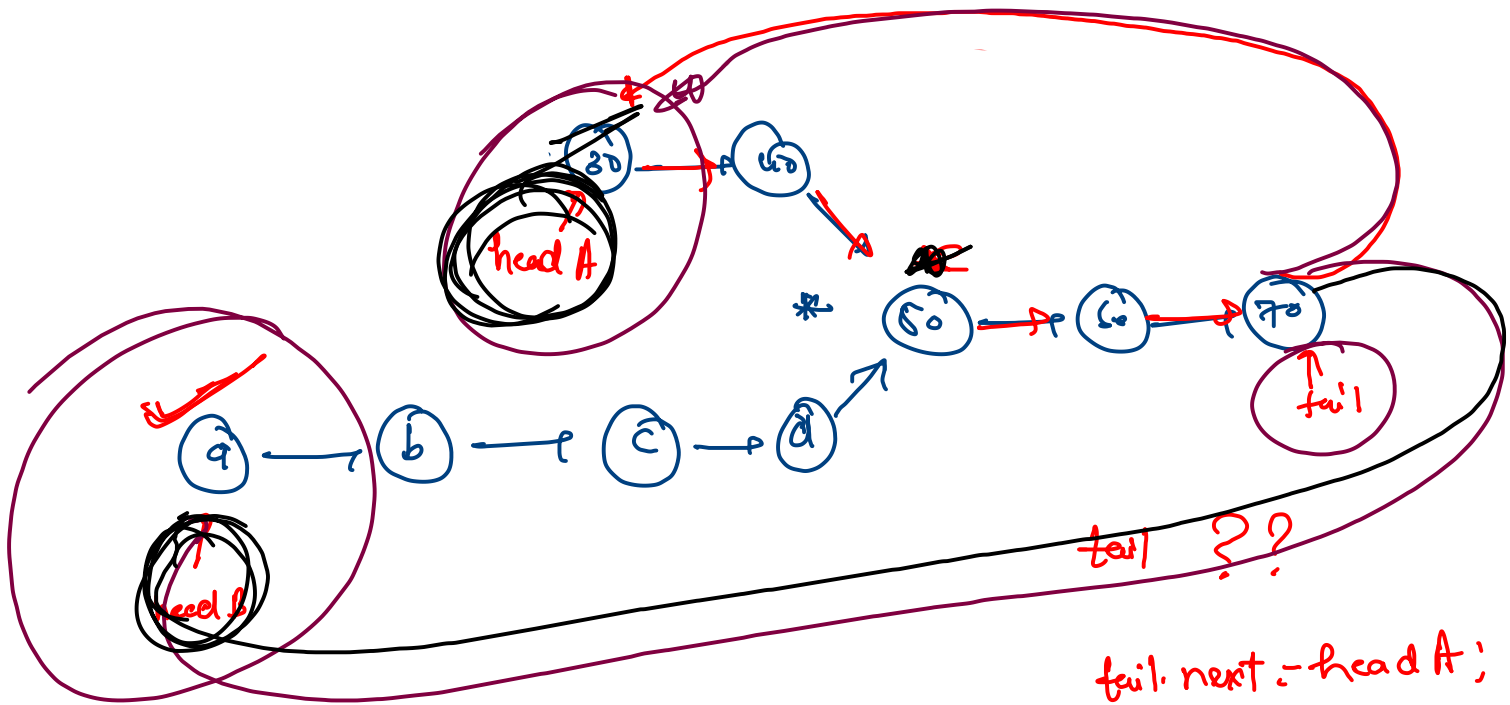
Intersection of List

Using difference Method: ⟶ all data are Integer



Intersection of LinkedList using Floyd cycle detection method :→

Detect using Cyclic Method

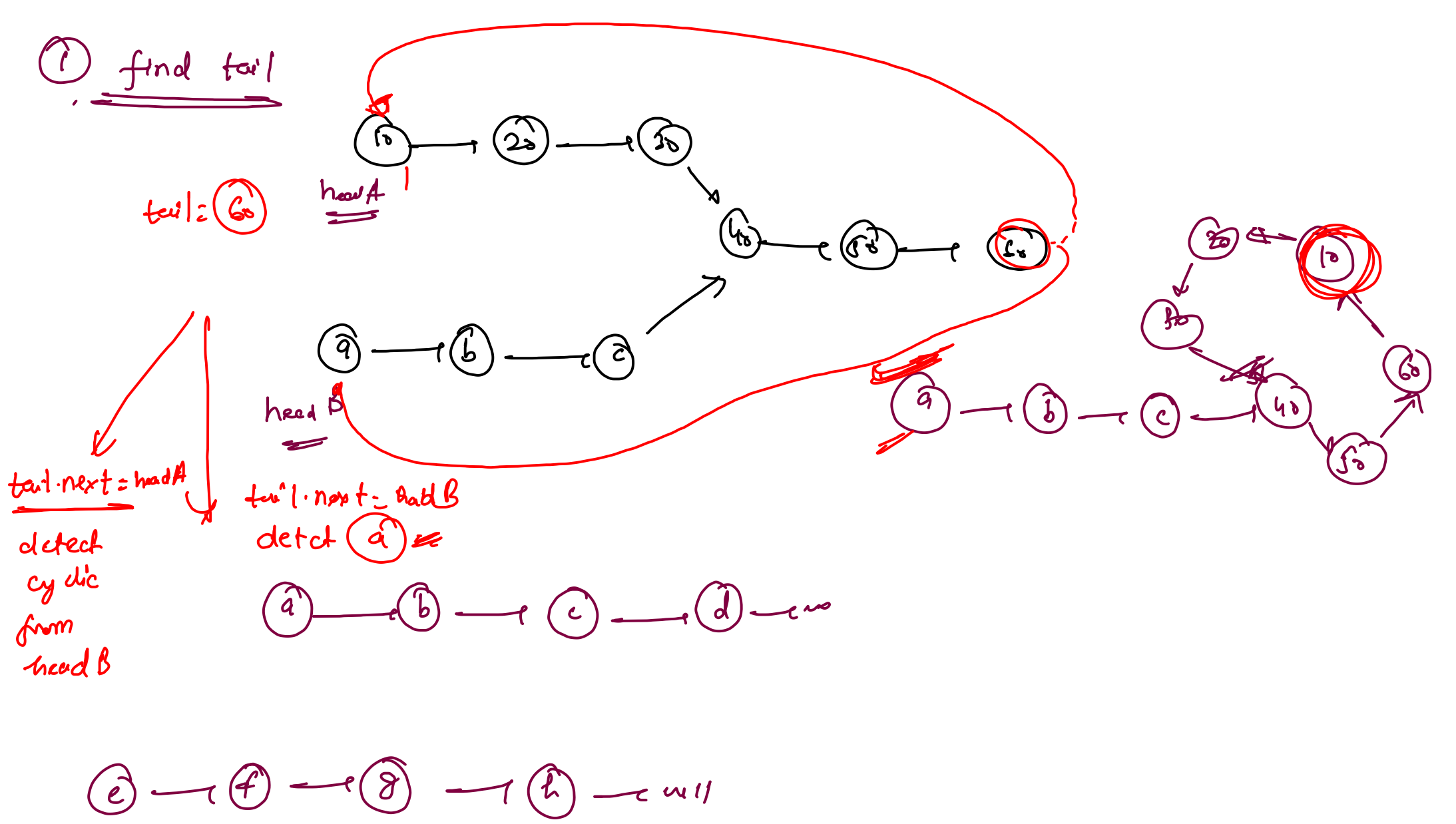


tail.next = headA;

check if (head B is cyclic)

TRUE → Starting of cycle

False → Return null


1 find tail
tail = 60
headA
10
20
30
40
50
60
a
b
c
headB
tail.next = headA
detect cyclic from headB
tail.next = headB
detect a
a
b
c
d
null
e
f
g
h
null