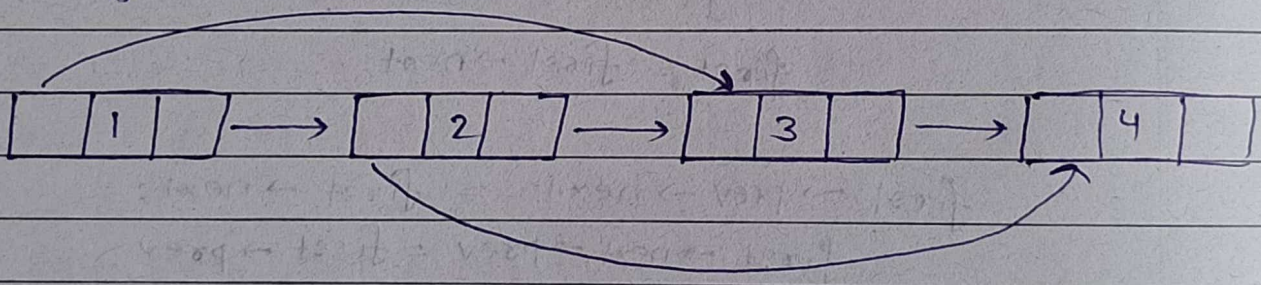
Lecture - 34

(Linked List - Hard)

* problem - 1

Clone a linked list With next and Random pointer :

→ The linked list is a type of doubly linked list.
→ One pointer is pointing to next Node.
→ Second pointer is pointer to any random Node.
→ It is Not Necessary that Every Node is pointing to a random Node.



Task :

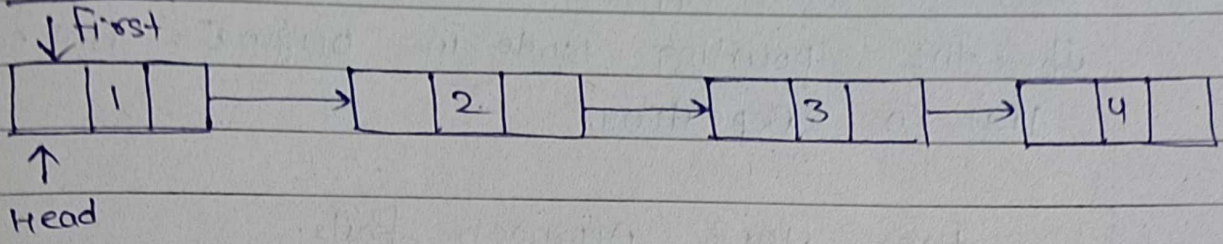
Copy the same linked list and return as it is.

↔ Approach 1

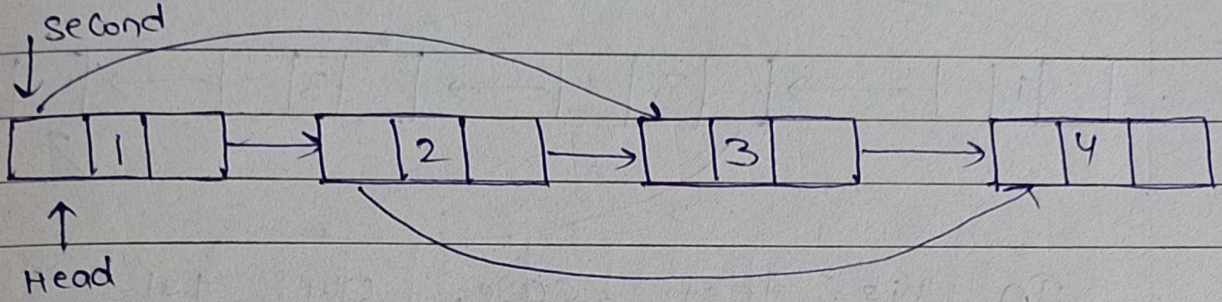
Make a duplicate linked list having only One pointer (Next pointer).

Page No.:
Date: / /

Duplicate:



Make original linked list also:



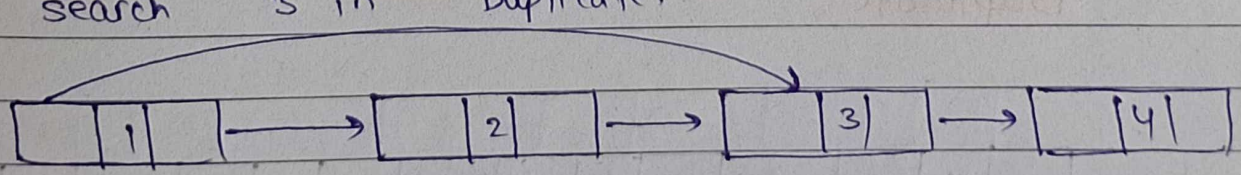
put Head on both linked list.

First pointer on duplicate linked list and second pointer on original linked list.

Check in original array that the first Node pointer to which Node which is Random.

We find 1 point to 3.

search 3 in Duplicate.



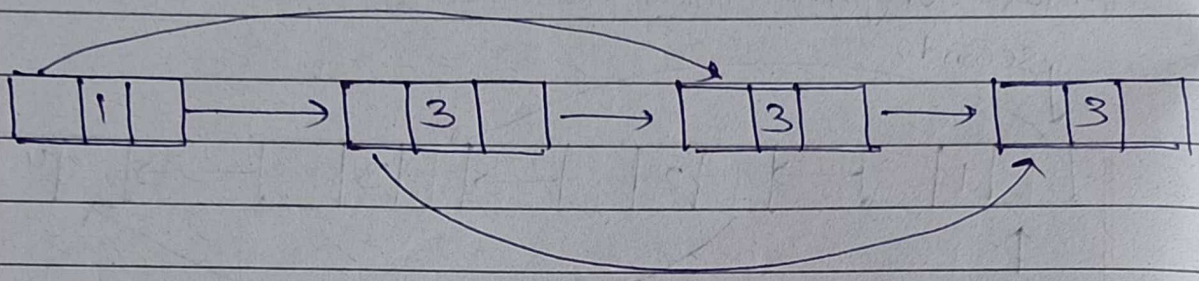
2 == 3 ⟶ 3 == 3
No Yes

point 1 in 3

Teacher's Signature.....

But, in this Approach we have a problem, if the pointing Node in original Linked list in repeating,

then this Approach fails.

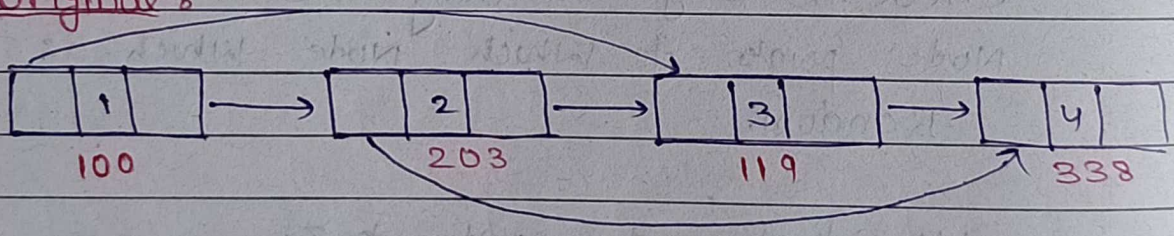


In this test case, our 1st Approach fail.

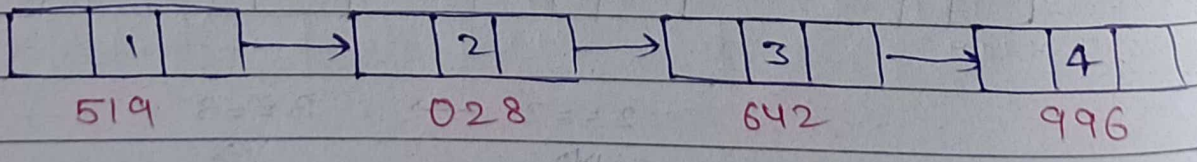
2nd Approach :

We have an another approach in which we can search by address.

Original :



Duplicate :



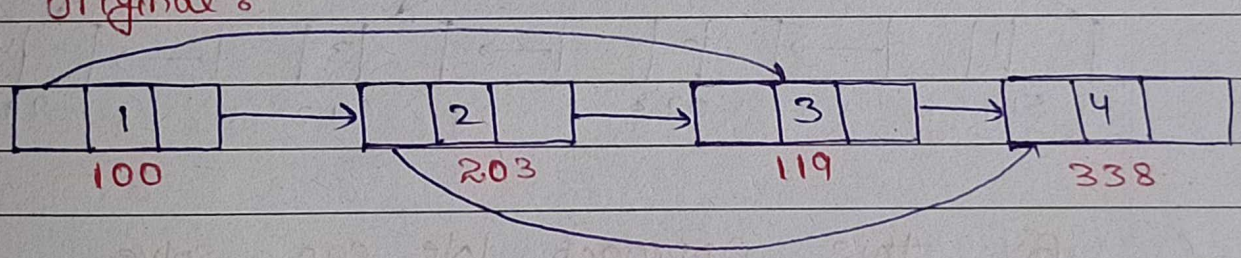
Teacher's Signature.....

We will store the address of original linked list and duplicate linked list with equivalent Nodes.

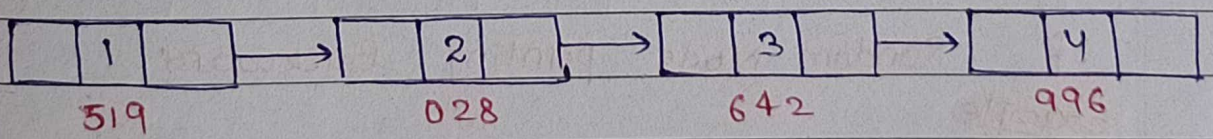
[100 → 519]
[203 → 028]
[119 → 642]
[338 → 996]

In original ~~array~~ linkedlist 1 → 3 and their address 100 → 119.

In duplicate linkedlist we traverse and check with the original array.



Duplicate:



Check in the Memory we
100 pointing to 119
find equivant address of duplicate
100 → 519
119 → 642

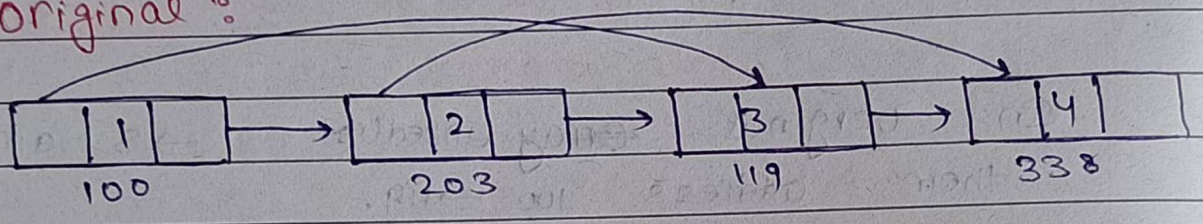
Teacher's Signature.....

① 519 → 642 (Random)

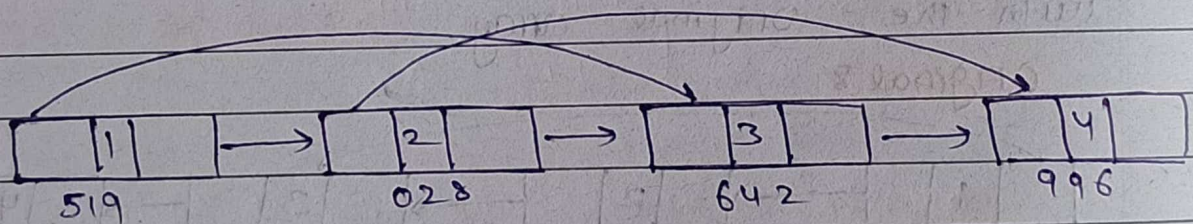
and 203 pointing to 338
Equivalent of 203 is 028
Equivalent of 338 is 996

⑪ 028 → 996 (Random)

Original :



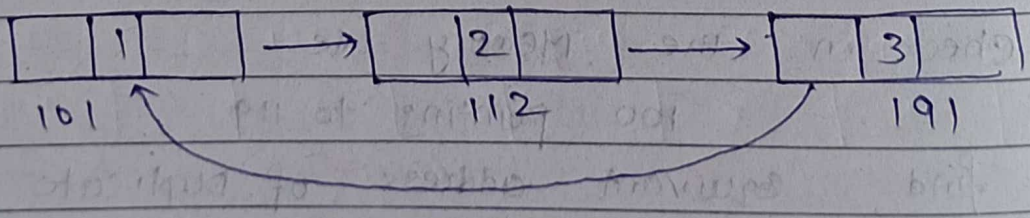
Duplicate :



By this approach we can solve the problem
for the Random Node pointing forward.

If Random Node pointing Backward

Example



Teacher's Signature.....

In this it is access the back Node. But dont traverse backward.

So, this Approach fails also if we have address.

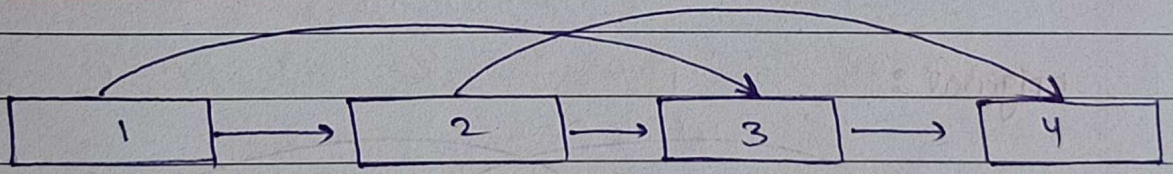
one More issue is

In this Approach, we have space Complexity is O(N).

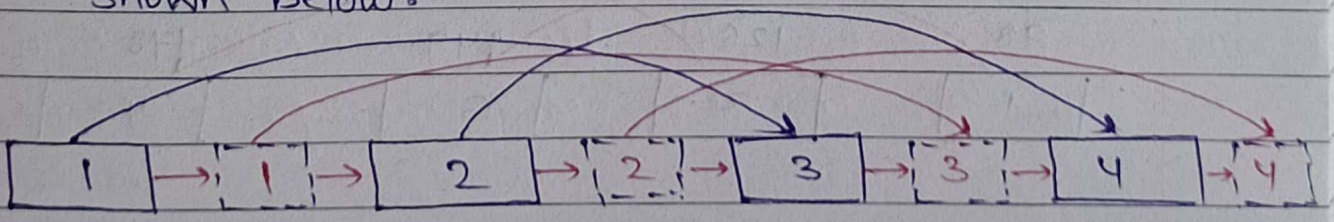
But we have to solve this in O(1). This Approach also fails.

We have to solve this Without using any Extraspace

Approach 3

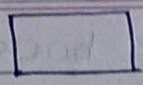


In this Approach, We will make a duplicate Node, Just after the Node which is present in original Node. Shown below:

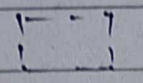


Like this.

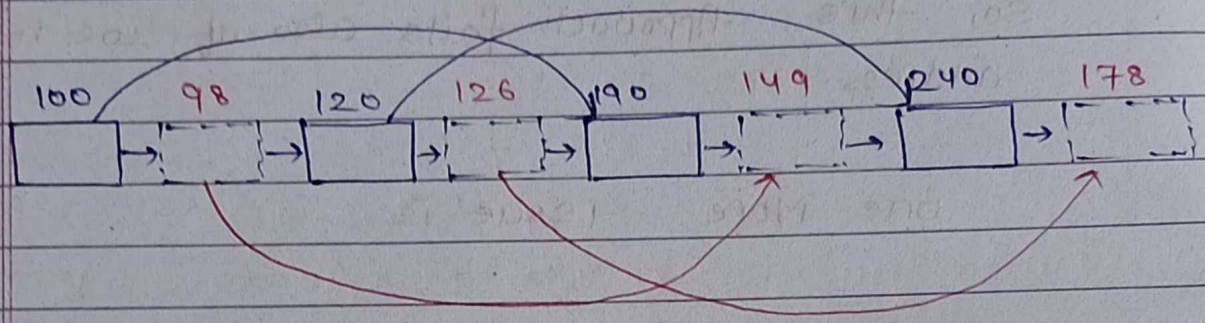
Teacher's Signature.....



→ original



→ Duplicate.



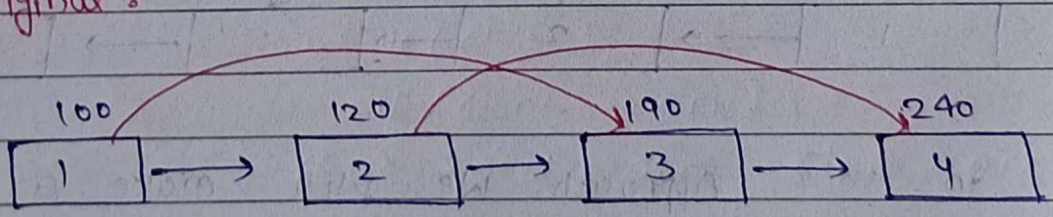
After this we will divide the original and duplicate linked list.

100 = 98
120 = 126
190 = 149
240 = 178

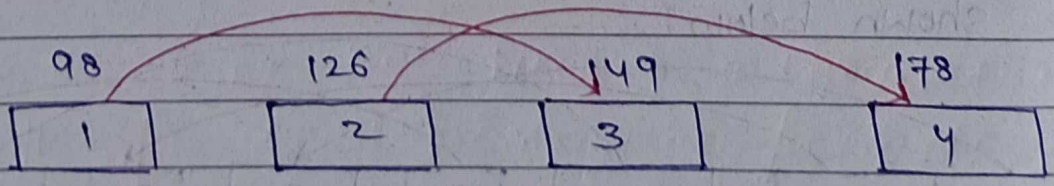
But we cant use this, we can done by next only.

Dividing the linkedlist which is Merged.

original :



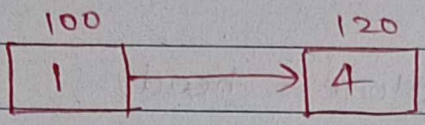
Duplicate :



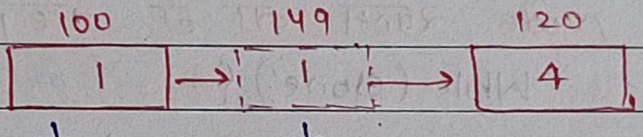
Teacher's Signature.....

How to code :

① हर Node के आगे उसका Duplicate Node copy कर के Attach kar denge



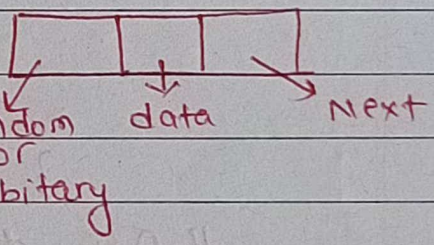
duplicating of 1



original

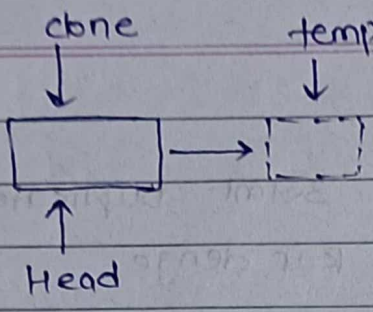
duplicate of 1.

// class of Node




```
class Node {
    public :
     int data;
     Node * next;
     Node * arbt;
    Node (int x) {        // Constructor.
        data = x;
        next = NULL;
        arbt = NULL;
    }
}
```

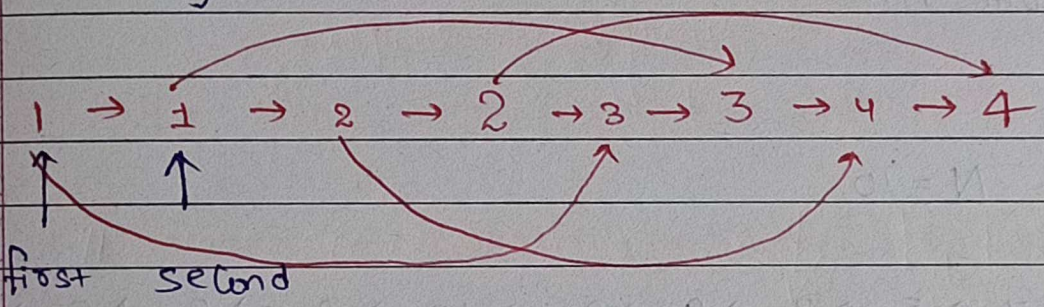
Teacher's Signature.....

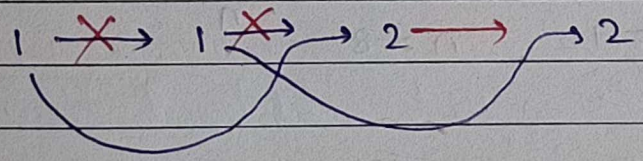



```
Node * clone = head;
Node * temp;
// duplicate कब तक होगा → जबतक original linked list
// का Node खत्म ना हो जाए।
While (clone) {
// Create New node
temp = new Node ( clone → data );
// temp अब clone के आगे जाएगा
temp → next = clone → next;
clone → next = temp;
Clone = clone → next → next;
}
// All duplicate Node are Created
clone = head;
// Now, We point the random pointer.
While (clone) {
if (clone → arb) {
Clone → next → arb = clone → arb → next;
}
clone = clone → next → next;
}
// Now we will break the linked list one
// for original and another for duplicate.
```

Teacher's Signature.....

```
Node * ans = head → next;
    clone = head;
temp = head → next;
While (temp) {
    clone → next = temp → next;
    clone = temp;
    temp = temp → next;
}
return ans;
}
```





1 → 2 ~

1 → 2

Page No.
Date : / /

* **Problem : 2**

Reverse a sublist of a linked list

You have given a linked list.

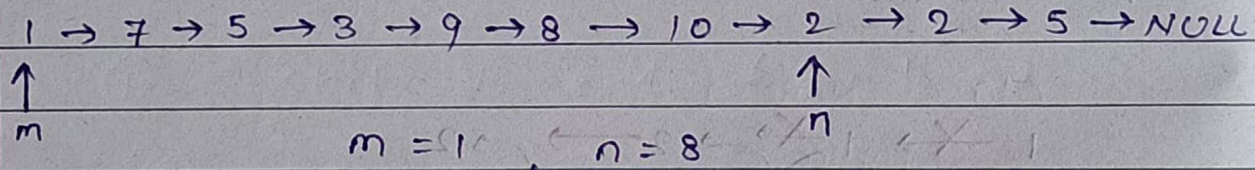
you have given 2 pointer m and n pointing 2 nodes in a linked list.

Task :

Reverse the Node present from m to the node n.

Ex :

N = 10



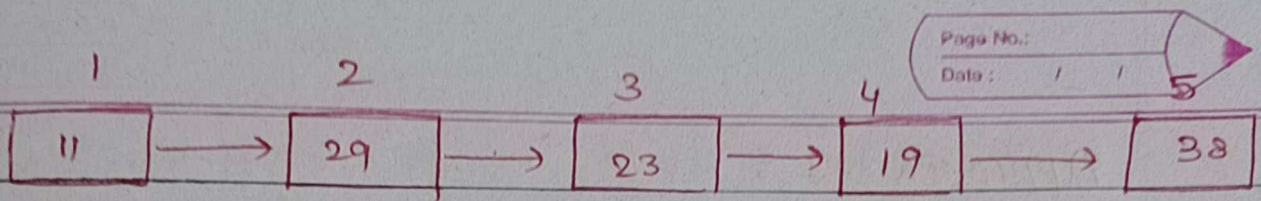
m = 1 , n = 8

Output :

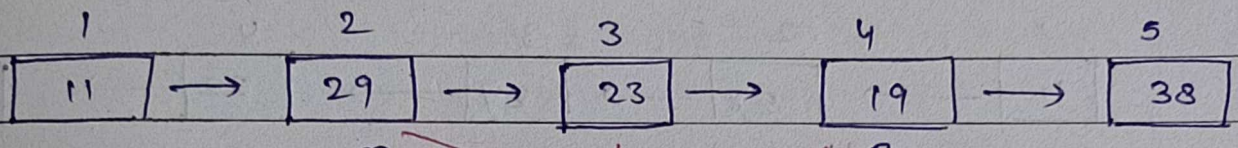
2 → 10 → 8 → 9 → 3 → 5 → 7 → 1 → 2 → 5

reversed

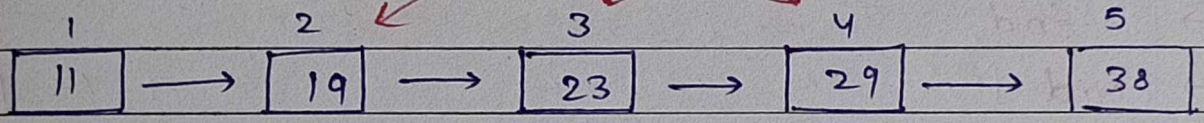
Teacher's Signature.....



$m = 2$
$n = 4$
} → Reverse the Node present inside m and n.



~~After~~ Reverse



We have to reverse the linked list without doing swap.

Approach 1

Copy the Node of linked list into an array.

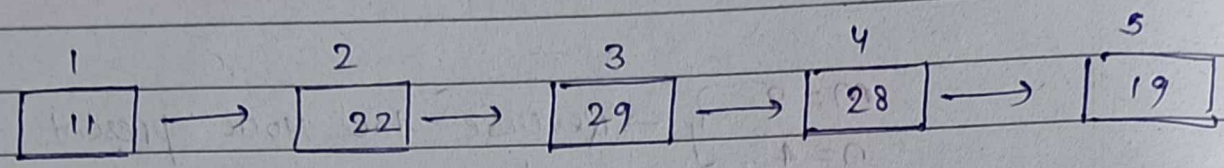
The reverse in array easily.

But we use here extra space and we don't have to use.

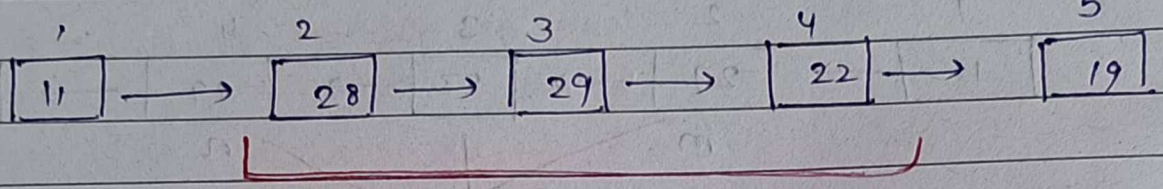
This Approach fails.

Teacher's Signature.....

Approach 2 :



m = 2
n = 4



→ Start
→ End
→ Mid

Here 4 cases Arisers :

① start to End (1 to 5)

② start to mid (1 to 4)

③ mid to end (3 to 5)

④ mid to mid (2 to 4)

We have to take care for all four cases.

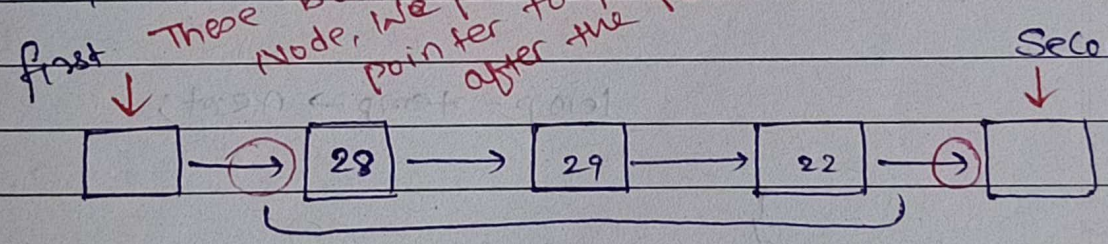
Teacher's Signature.....

Find How many No. We have to reverse :

→ From Where We start

```
Node * Reverse ( Node * head, int n) {
    Node * prev = NULL;
    Node * prev = temp;
    While ( n--) {
        temp = head → next;
        head → next = prev;
        prev = head;
        head = temp;
    }
    return prev;
}
```

After Reverse :



We have to point the reverse part With previous part or last part.

first and second ka address store karwanta tha or phir reverse ke baad point karr dena hai.

Teacher's Signature.....

13 → 21 → 15 → 17 → 30 → 28 → 91 → 40

reverse part

m = 3 n = 7

```
First = NULL;
Second = NULL;
Node * temp = head;
int Count = 1;

while ( Count <= n+1) {

// here address of 3rd & 7th Node are stored in first & second.

if (Count == m+1)
    first = temp ;
if (Count == n+1)
    second = temp;

temp = temp → next;

}
```

→ Cases:

```
if ( First == NULL){ // started from 1st Node

if (second == NULL){ // ended to last node.
    return Reverse(head,5); // n
}
else {                  // first to middle;
temp = Reverse (head, 3); // n
head → next = second;
return temp;
}
```

Teacher's Signature.....

// If not started from 1 Node
// started for middle to end & middle to middle.

```
                // Middle to end.
        if (second == NULL) {   // second NULL
first->next = Reverse(first->next, n-m+1);
        return head;
        }

        // Middle to Middle
        temp = first->next;
first->next = Reverse(First->next, n-m+1);
        temp->next = second;
        return head;
```

1 → 2 → 3 → 4 → 5 → 6 → 7

4 case :

case ① Start to end (Total)

Reverse ही Return होगा।

case ② Start to Midd.

Reverse + Some last element

```
temp = Reverse(head, n)
    head->next = second;
    return temp;
```

Teacher's Signature.....

Page No.
Date : / /

Case (3) : Mid to end

some element + Reverse

~~int temp;~~

first → next = Reverse (first → next, n-m+1)

Case (4) Mid to Mid

some Element + Reverse + some element

temp = first + next;

first next = Reverse (first → next , n-m+1)

temp → next = second;

return head;

Teacher's Signature.....