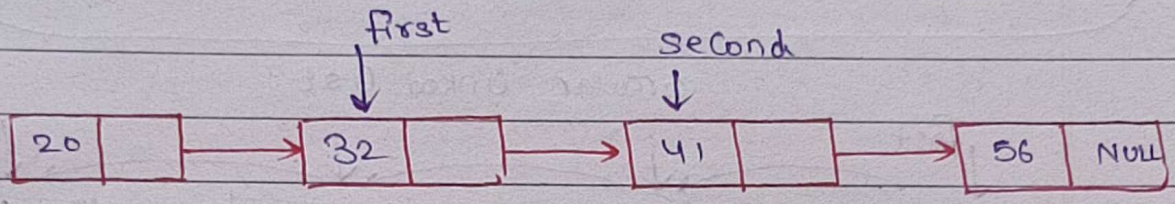
Lecture - 32

(Linked List {Problems})

If you want to delete a Node in Linked list.



Delete 41

Approach 1 :

point the Node of 32 to direct 56
41 is not linked to anyone.
so, it can't be shown.

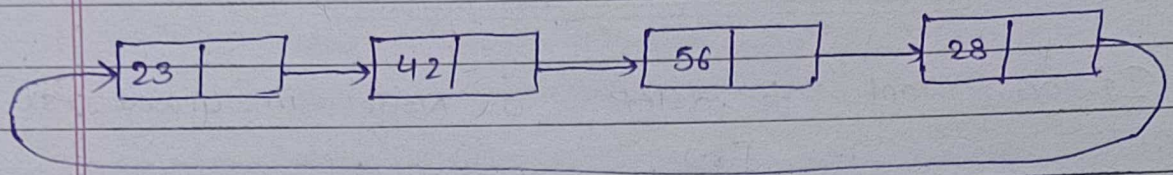
first → next = first → next → next

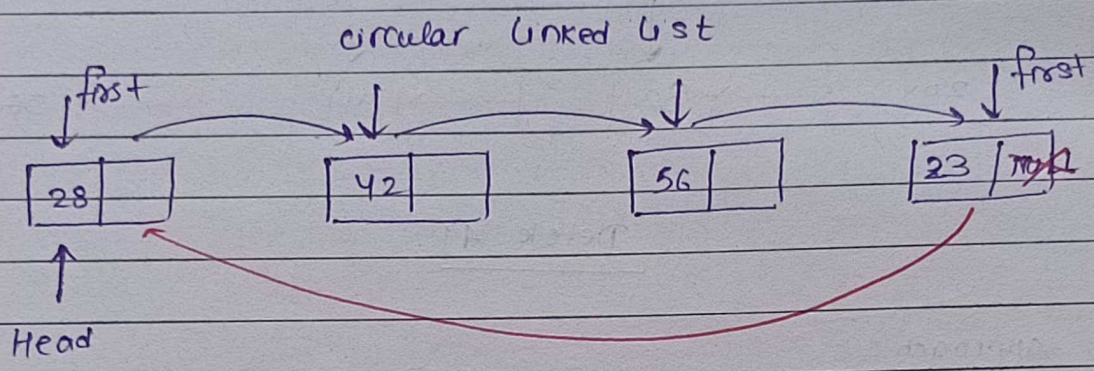
Approach 2 :

put a second pointer on the node which we want to delete. and after we do.
delete (second);

Teacher's Signature.....

Circular Linked List

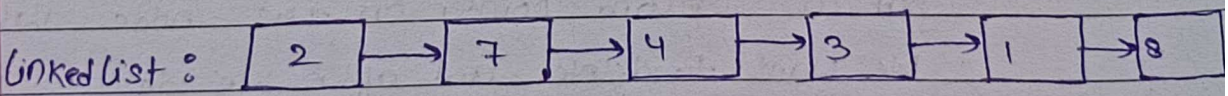





```
Node * first = Head
while( first → next){
    first = first → next
}
first → next = Head;
```

Array :

0	1	2	3	4	5
2	7	4	3	1	8



Accessing 5th element in Array:

O(1) arr[4] = 1

In linked list

O(N) Manually Checking one - one element

Teacher's Signature.....

But Insertion and deletion can be easy in linkedlist.

Array : T.C : O(N) At start

Linked list : T.C : (O(1) At start

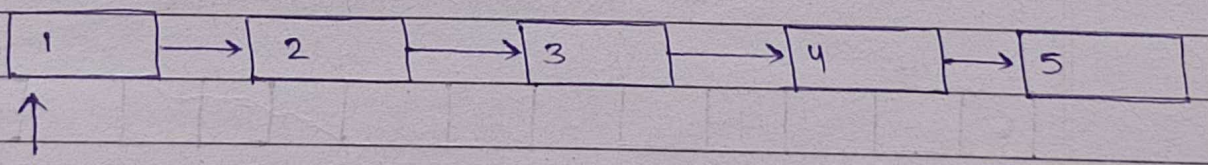
Searching:

Array (if sorted) : O(log n)
↳ Binary search

Linkedlist (if sorted) : O(n)

problem 1: Check if Circular Linked list

Given, a linked list With head.



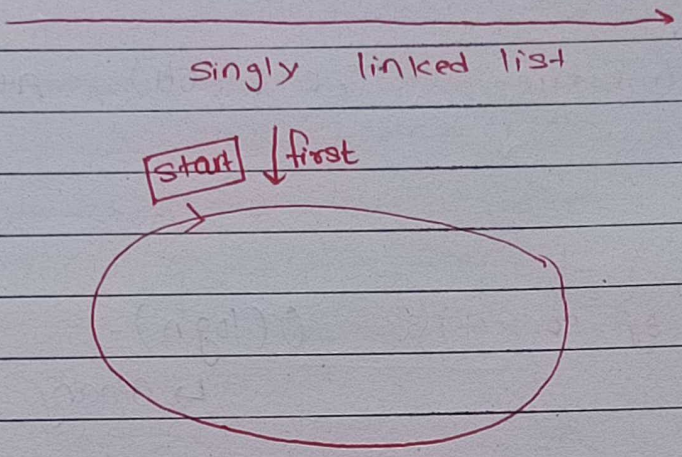
Approach 1:

If NULL present in linkedlist, then the linked list is not circular.

Traverse the whole array.

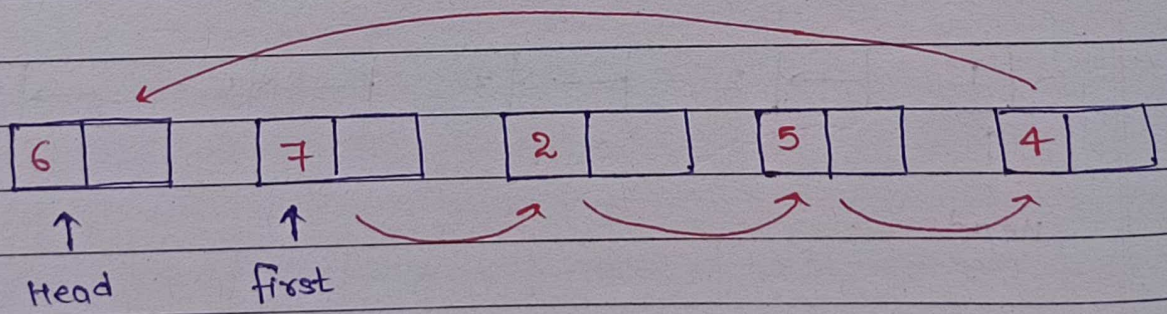
Teacher's Signature.....

But here is a problem, if linked list is very long then it create problem.



Circular linked list

Fix the start at first Node.
Make an another pointer called 'first' and traverse, if start found then it is Circular.



First can find either NULL or Head.

If NULL
then singly

If Head
then Circularly.

Teacher's Signature.....

first pointer को घुमाते घुमाते या तो NULL मिलेगा या start/Head मिलेगा।

अगर NULL मिला तो
↳ circular नहीं होगा।

अगर start मिल गया तो
↳ circular हो जाएगा

Code :

```
bool isCircular (Node * head) {
    Node * first = head;
    while (first → next && first → next != head) {
        first = first → next;
    }
    if (first → next == NULL)
        return 0;
    else
        return 1;
```

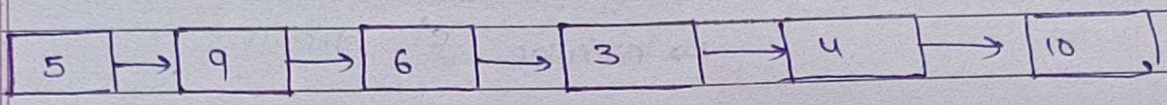
Page No:
Date: / /

Problem-2

Find the Sum of Last N nodes of Linked list:

A single linked list of size M,

Return the sum of last n nodes.



Total Size = M = 6

n = 3

Sum of last 3

We have to neglect some Nodes from starts.

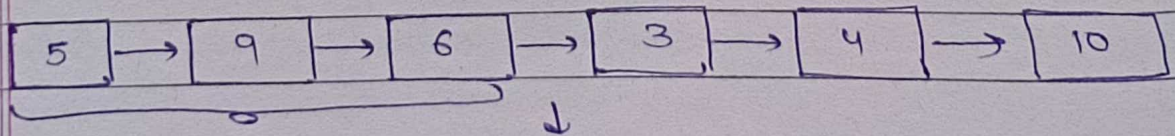
Total No. of Neglected Nodes, a

= M - n

= 6 - 3

= (3)

↳ We have to neglect 3 nodes.



Neglect a = 3 Nodes

Sum = 0

0 + 3 + 4 + 10

= 17

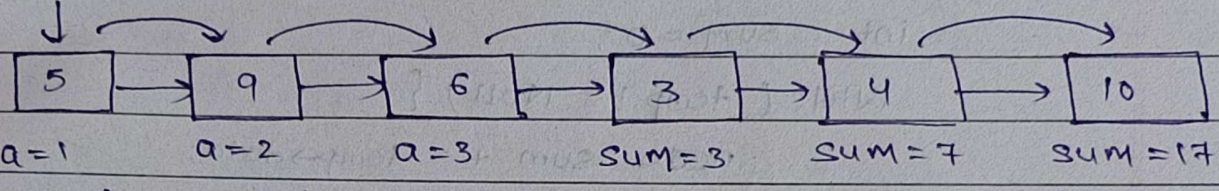
return 17.

Teacher's Signature.....

Page No.
Date : / /

put a pointer on first Node. First traverse 'a' Nodes without adding element.

Then sum all the element after a = 3 Nodes.



step-1 : Count total Number of Nodes

step 2 : Find the no. of Node for Neglecting.

step-3 : A temp pointer can neglect first some nodes while traversing.

step 4 : The start Adding the element.

Code :

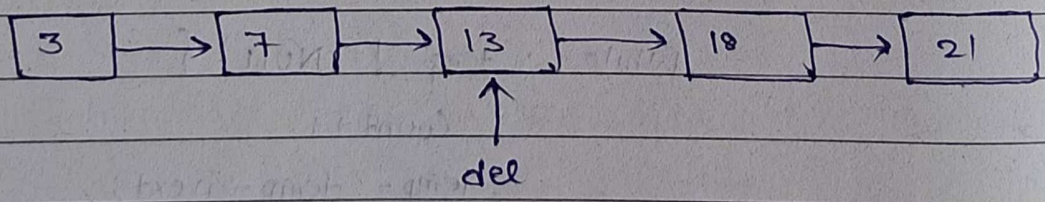
```
int SumofLastN_Nodes (struct Node* head, int N){
    int Count = 0;
    Node* temp = head;

    While ( temp != NULL) {
        Count++;
        temp = temp->next;
    }

int neglect = Count - N;

temp = head;
```

Teacher's Signature.....

```
while ( neglect > 0) {
        temp = temp -> next;
        neglect--;
    }

int sum = 0;
while ( temp != NULL) {
        sum += temp -> data;
        temp = temp -> next;
    }

return sum;
}
```

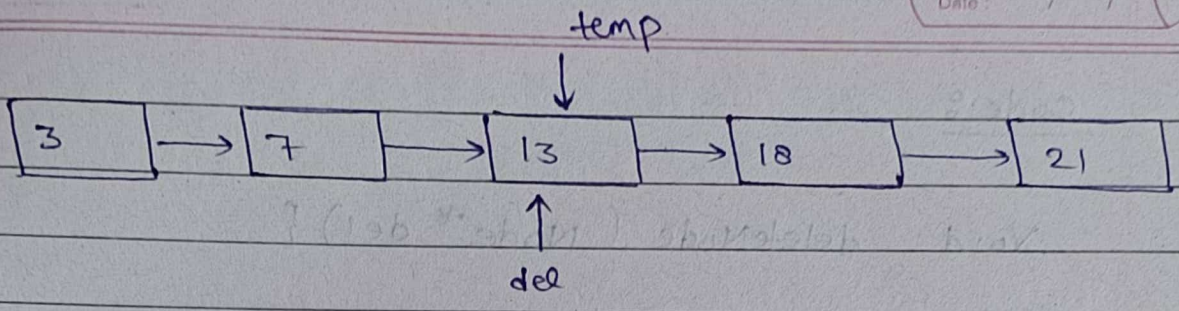
* **Delete Without head pointer :**

A linked list
↳ you only have given a pointer of node which you have to delete.



You have give a del pointer pointing to a node.
Delete that node.

Teacher's Signature.....

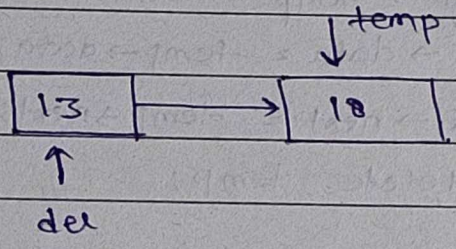


Make a pointer temp at same Node of del.

```
Node * temp = del;
```

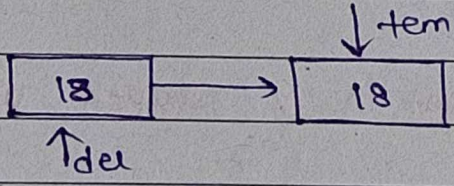
Move one step the temp pointer.

```
temp = temp → next;
```

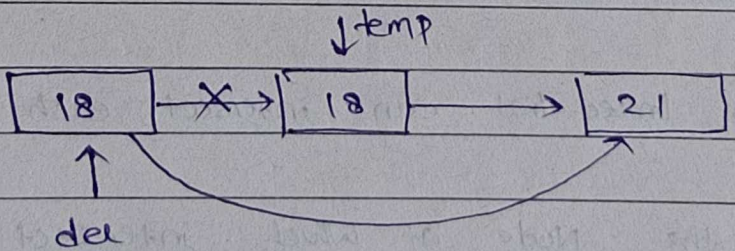


copy the data of temp in del

```
del → data = temp → data
```



del में temp के Next Node1 का pointer / address Point करो।



~~नहीं~~ del → next = temp → next.

At last delete the temp.

```
delete temp;
```

Teacher's Signature.....

Code :

```
Void deleteNode (Node* del) {

    if (del == NULL || del -> next == NULL)
        return;

    ~~del * temp~~
    Node * temp = del;
        temp = temp -> next;
        del -> data = temp -> data;
        del -> next = temp -> next;
        delete temp;
    }
};
```

* Problem - 4

Intersection point in Y shaped linked list.

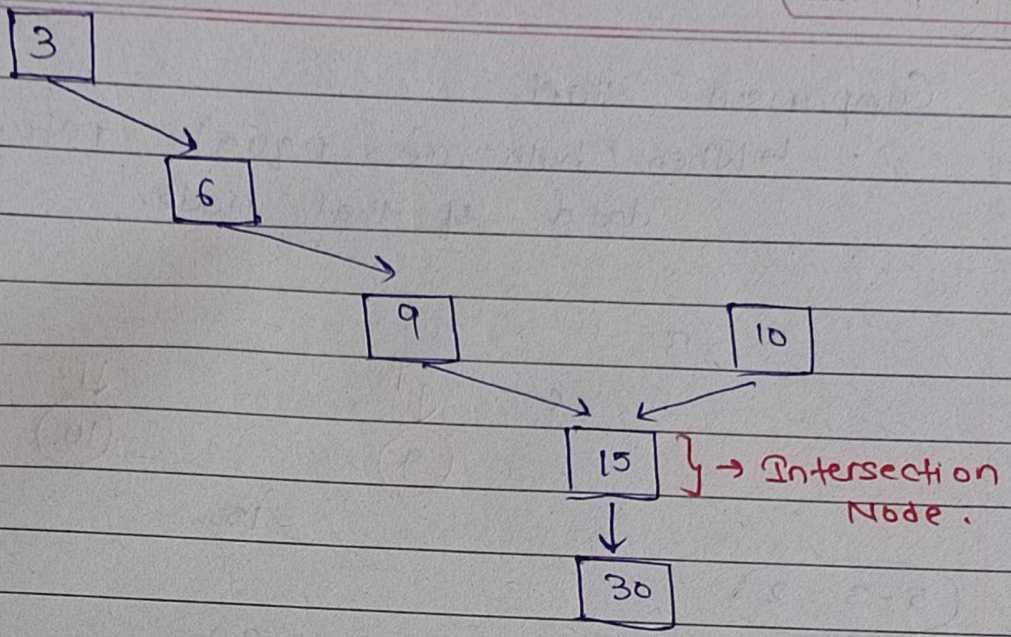
Given : Two single linked list
Both size are different.

Both linked list can intersect each other

Find the Node at which intersect.

Two linkedlist :
Linkedlist1 : Size → N
Linkedlist2 : Size → M

Teacher's Signature.....



How to find:

Check Both Linked list.

↳ For this the size should be same.

Make the size equal

Bigger linked list ko small karenge.

Size 1 = 5

Size 2 = 3

5 - 3 = (2)

↓

Linkedlist1 2 Node ke baad
Ek pointer lega और
Linkedlist2 first Node pe Ek
pointer लेगा।

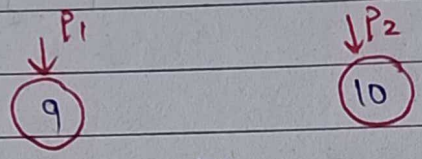
Teacher's Signature.....

Comparison start

↳ When both are equal return the data of that node.

3

6



15

5 - 3 = 2

30

9 10

P1 != P2 (True)

P1 = P1 → next;

P2 = P2 → next;

15 15

P1 == P2 (True)

return P1 → data

15

Code:

```
int intersectpoint ( Node * head1, Node * head2) {

// step 1 : calculate the length of Both linked list
    int len1 = 0, len2 = 0;
    Node * temp1 = head1;
    Node * temp2 = head2;
```

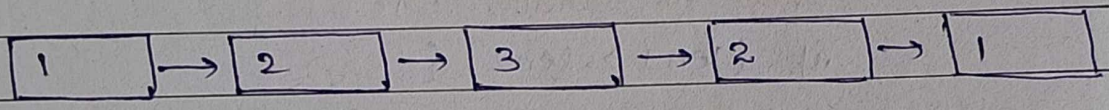
Teacher's Signature.....

```
while ( temp1 != NULL) {
        len1++;
    temp1 = temp1 → next;
}

while ( temp2 != NULL) {
        len2++;
    temp2 = temp2 → next;
}

// step 2 : Make longer linked list smaller as same
//          size to smaller one.
if (len1 > len2) {
    int diff = len1 - len2;
    while ( diff > 0) {
        head1 = head1 → next;
        diff--;
    }
else {
    int diff = len2 - len1;
    while (diff > 0) {
        head2 = head2 → next;
        diff--;
    }
}

// step 3 : start Comparing both linked list
//      if Matched return data otherwise -1.
```

Teacher's Signature.....

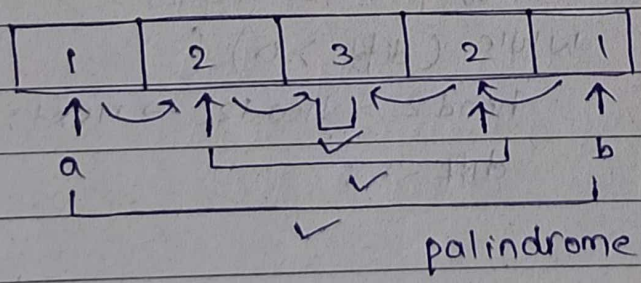
```
While ( head1 != NULL && head2 != NULL ) {
        if ( head1 == head2 ) {
            return head1 -> data;
        }
        head1 = head1 -> next;
        head2 = head2 -> next;
    }
    return -1;
}
```

* Check if linked list is palindrome :



Approach 1 :

Linked list को Traverse कर के Array में store कर लो उस के बाद palindrome check कर लो।



T.C : O(n)

S.C : O(n)

↳ But we want O(1).

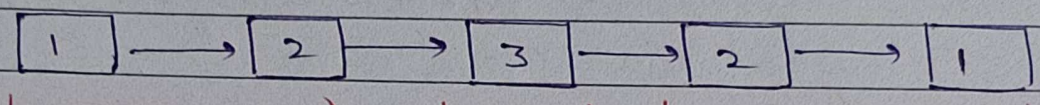
Teacher's Signature.....

Page No.
Date : / /

Approach 2:

Size of linked list is
either odd or even.

If odd :



first last.

Size = 5

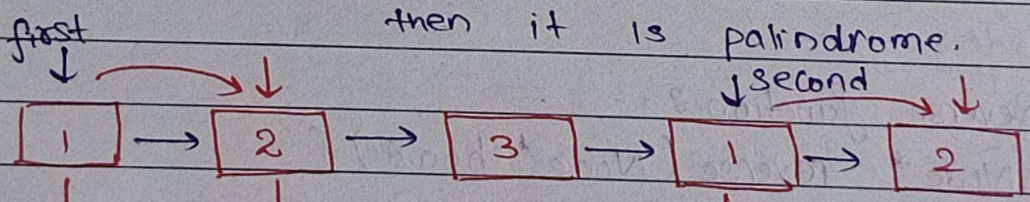
Divide in two parts.

$n = 5/2 = 2$

2 from first & 2 for last

Reverse the last part

If First is equal to second
then it is palindrome.



Take two pointer equal equal

first on Head

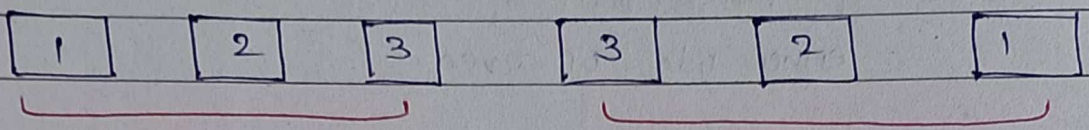
second on size - n

5 - 2 = 3 + 1 = 4

palindrome.

Teacher's Signature.....

If size is even.

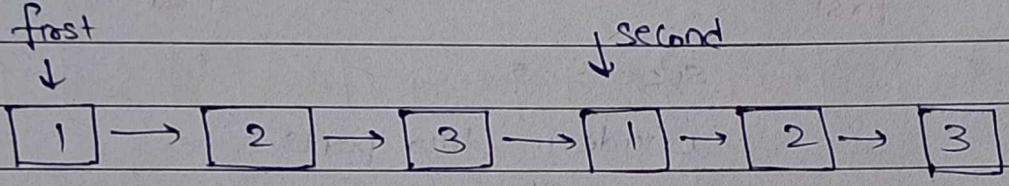


first second.

size = 6

n = 6/2 = 3

reverse second.



2nd pointer on → size - n + 1

Traverse and check.

Code:

Reverse function:

```
Node * reverse (Node * head) {
    Node * prev = NULL;
    Node * current = ~~prev~~ head;
    Node * next = NULL;

    While ( current != NULL) {
        next = current → next;
        current → next = prev;
        prev = current;
        current = next;
    }
    return prev;
}
```

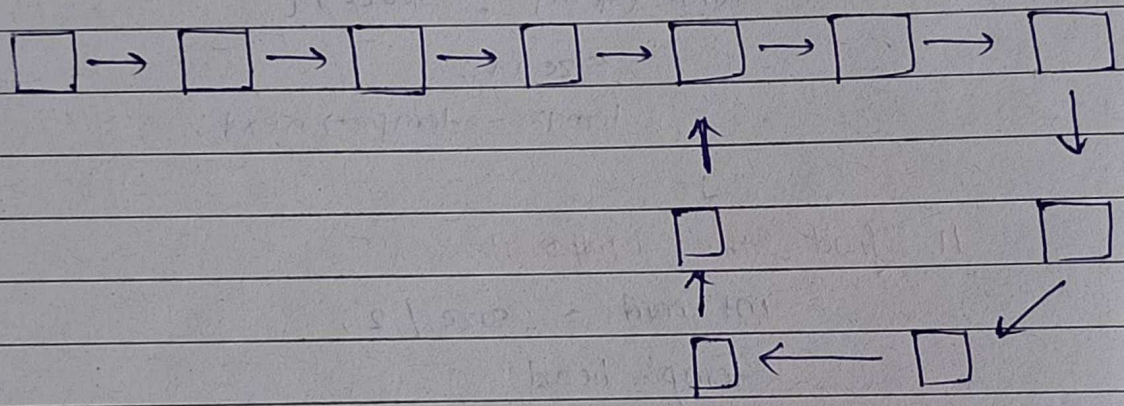
Teacher's Signature.....

```
bool ispalindrome (Node* head) {

    if (head = null || head -> next == NULL)
        return true;


// size of the linked list
        int size = 0;
        Node * temp = head;
        While (temp != NULL) {
            size++;
            temp = temp -> next;
        }
// find the Midpoint
        int mid = size / 2;
        temp = head;
// Reverse the second half of linked list
        ~~Node * head~~
        Node * prev= NULL;
    for (int i = 0; i < mid; i++) {
            prev = temp;
            temp = temp -> next;
        }
    prev -> next = reverse (temp);

// Comparing the both part of linked list.
        temp = head;
    Node* second = prev -> next;
for (int i= 0; i<mid; i++) {
    if (temp -> data != second -> data)
        return false
```

Teacher's Signature.....

```
        temp = temp → next;
        second = second → next;
    }
    return true;
  }
};
```

* Check that loop present or Not



How to solve:

Two Condition
- ↳ loop present
- ↳ loop Not present.

How to Approach:

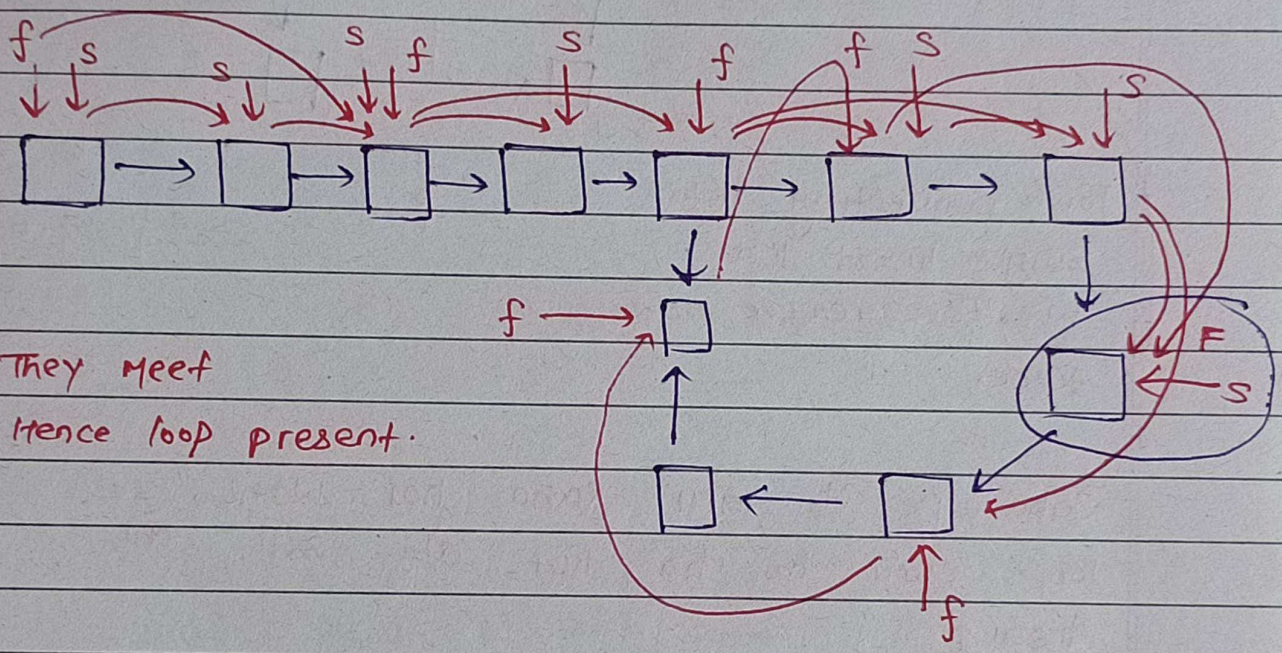
Take two pointer at first (Head) Node.

1st pointer can traverse 2 node at a time
2nd pointer can traverse 1 node at a time.

Teacher's Signature.....

If In the path, they meet NULL
↳ the linked list is Not having any loop.

If both pointer meet again excet NULL.
↳ the linked list in having loop.



They Meet
Hence loop present.

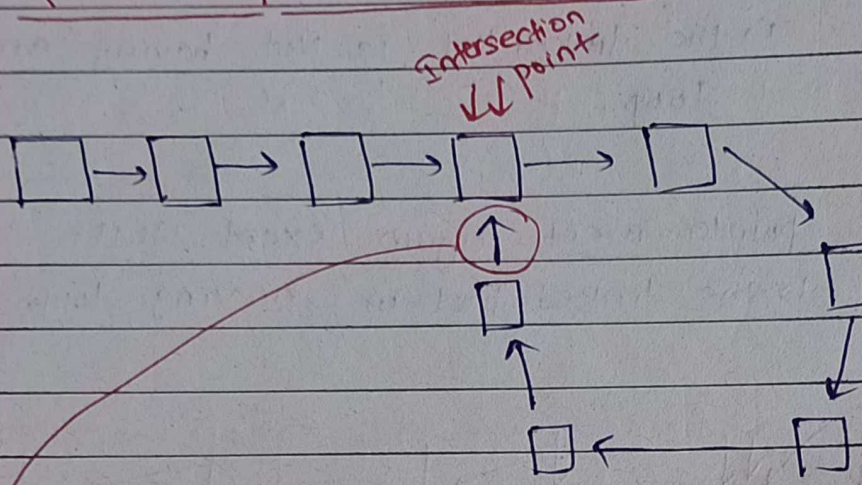
They Meet if NULL Not present.

They either Meet after n number of round.

Take Example of Race in Real life.

Teacher's Signature.....

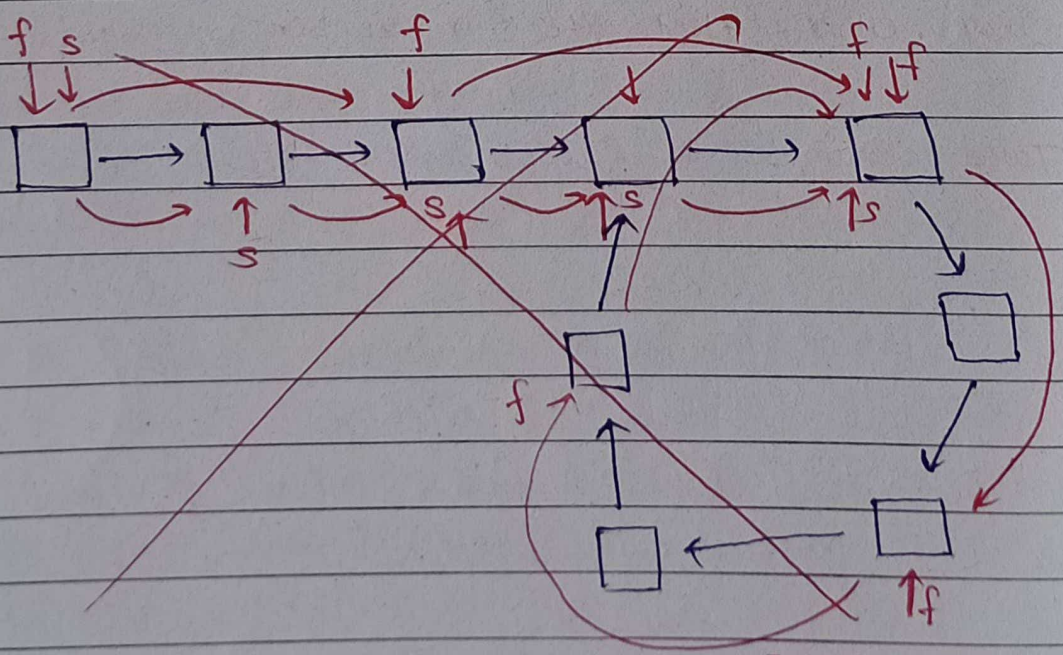
* Remove loop in Linked list :



For Converting into Single linked list we have remove this Node.

Jaha pe do Mill Raha hai Wahi se loop suru ho rha hai तो उसी को तोड़ना है।

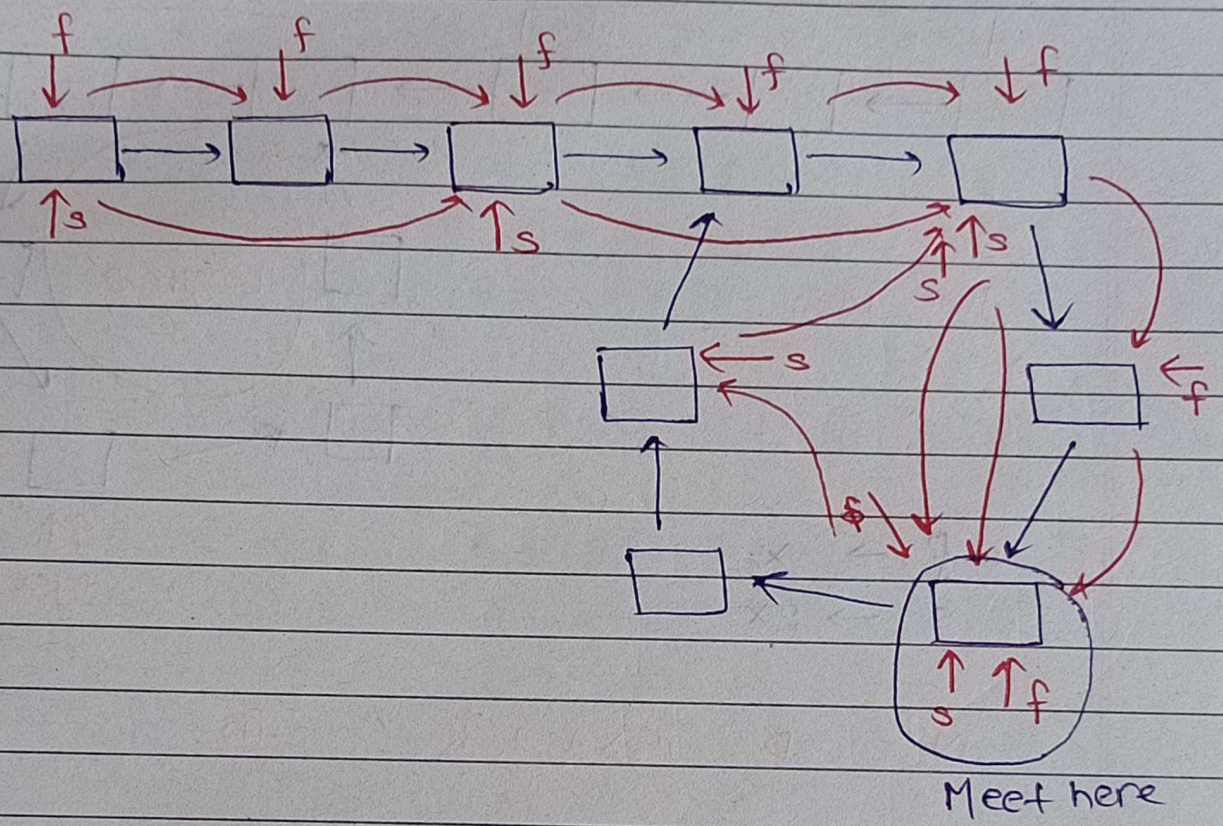
हमें ये कैसे पता चलेगा कि हमेशा दोनो उसी Node पे मिलेगा।



Teacher's Signature.....

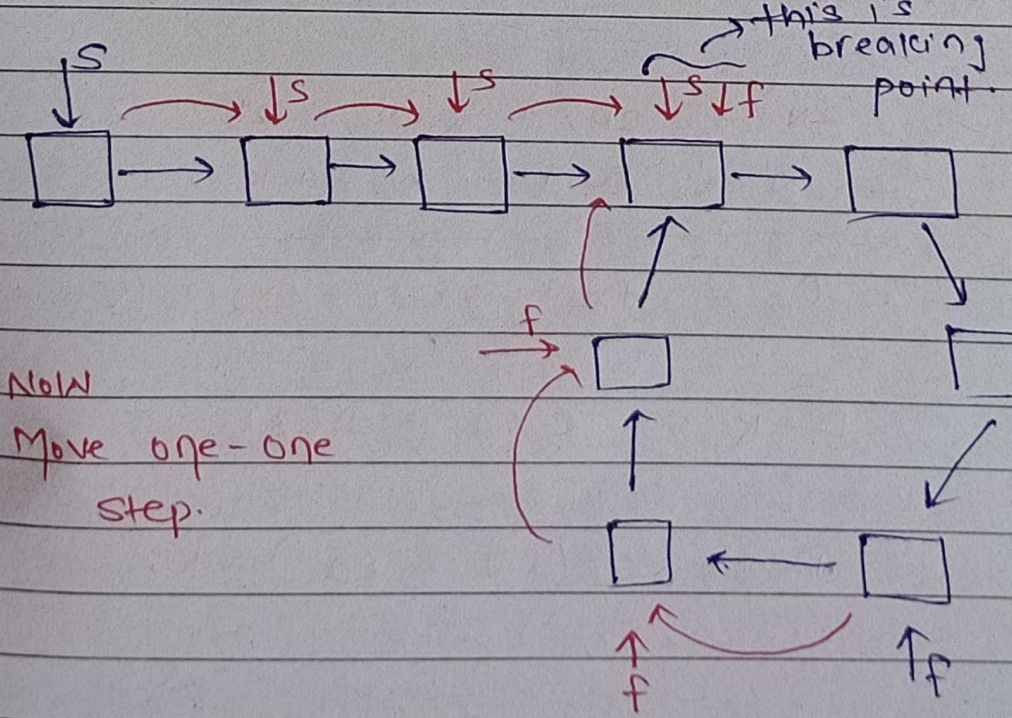
पहले First → 1 step

फिर Second → 2 step



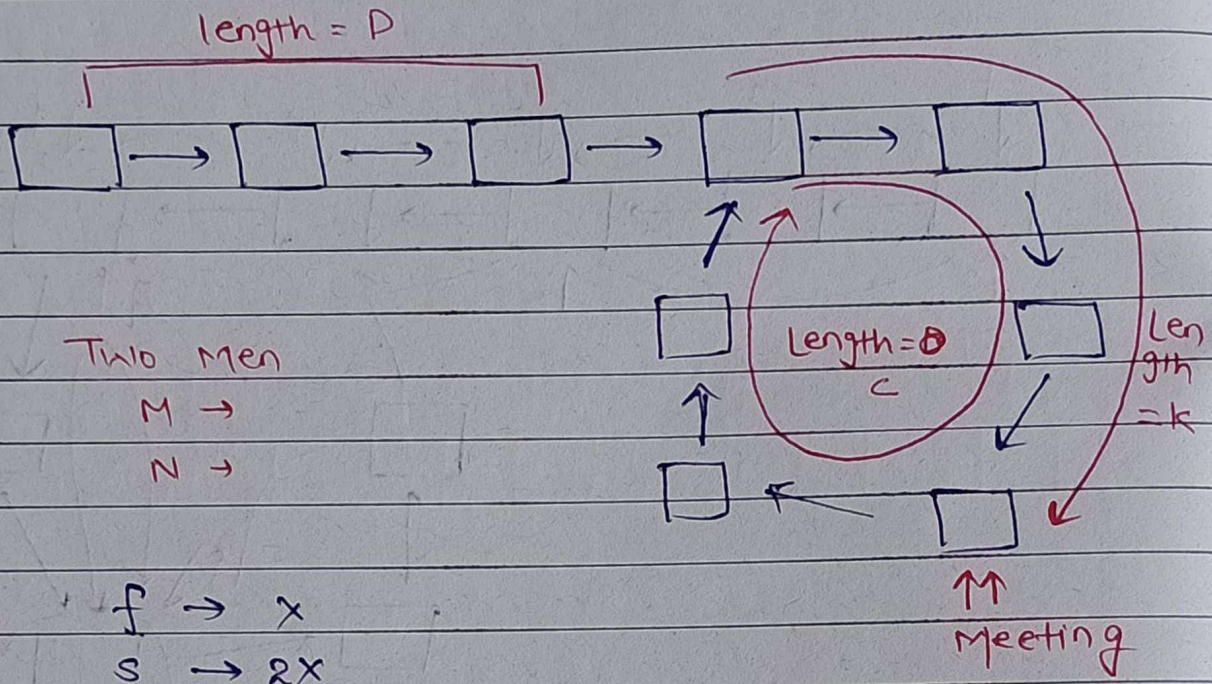
Send Second to head Node

first to at place.



Teacher's Signature.....

अब हम समझेंगे ये हुआ कैसे
ये एक Mathematics proof है।



$N = D + C \times i + K \quad -(1)$

$M = 2N = D + C \times j + K \quad -(2)$

→ Multiply by 2 → (1)

$$2N = 2D + 2C \times i + 2K$$
$$-\ 2N = D + C \times j + K$$
$$0 = D + 2C \times i - C \times j + K$$

$$D + K = C \times j - 2C \times i$$

$$D = C(j - 2i) - K.$$

Teacher's Signature.....

Page No.
Date : / /

Code :

```
Node *ptr1 = head;
Node *ptr2 = head;

While ( ptr2 != NULL && ptr2 -> next != NULL) {
        ptr1 = ptr1 -> next;
        ptr2 = ptr2 -> next -> next;

    if ( ptr1 == ptr2) {
        unsigned int Count = 1;
        While (ptr1 -> next != ptr2) {
            Count++;
            ptr1 = ptr1 -> next;
        }
        ptr1 = head;
        ptr2 = head;

    for ( int i = 1; i <= Count ; i++ ) {
            ptr2 = ptr2 -> next;
    }
    While( ptr1 != ptr2) {
        ptr1 = ptr1 -> next;
        ptr2 = ptr2 -> next;
    }

    While ( ptr2 -> next != ptr1 ) {
            ptr2 = ptr2 -> next;
    }
    ptr2 -> next = NULL;
    }
}
```

Teacher's Signature.....