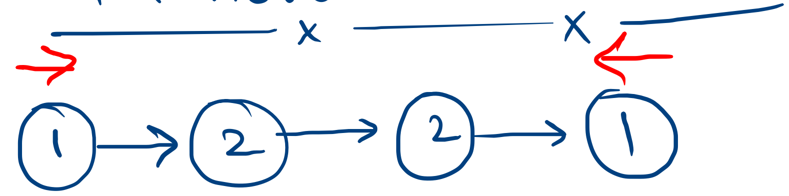
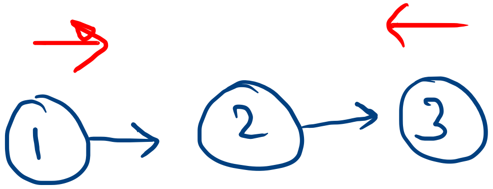
Palindrome Linked List

I/p :



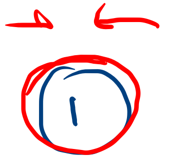
O/p : True

I/p :



O/p : False

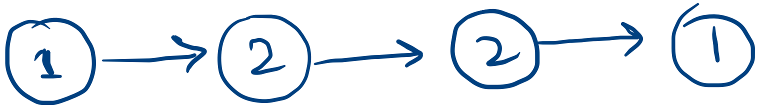
I/p :



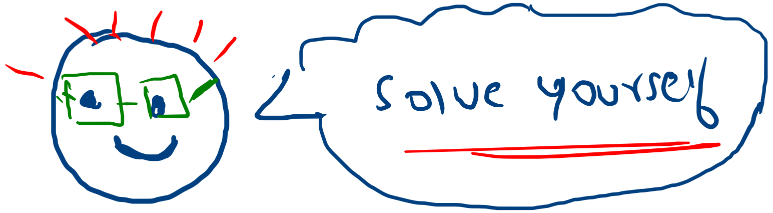
O/p : True

NAIVE Approach

I/p:



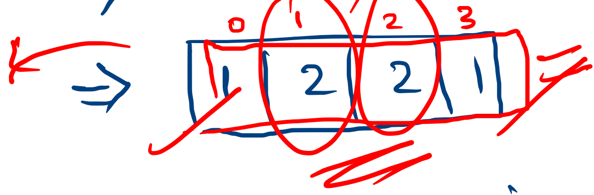
solⁿ :



NAIVE Approach

I/p: 1 → 2 → 2 → 1

sol^n: 1) Start traversal of List and store data in Vector



n=4

0 ↔ 1

i=0 ↔ 4-0-1
3

i=1 ↔ 4-1-1
2

2) check data is Palindrome or not?

```
for( int i=0; i< vec.size()/2 ; i--)
{
    if( vec[i] != vec[n-i-1]
        return false;
}
return True;
```

n = vec.size()

O(N) - Space
O(N) - Time

Better Approach

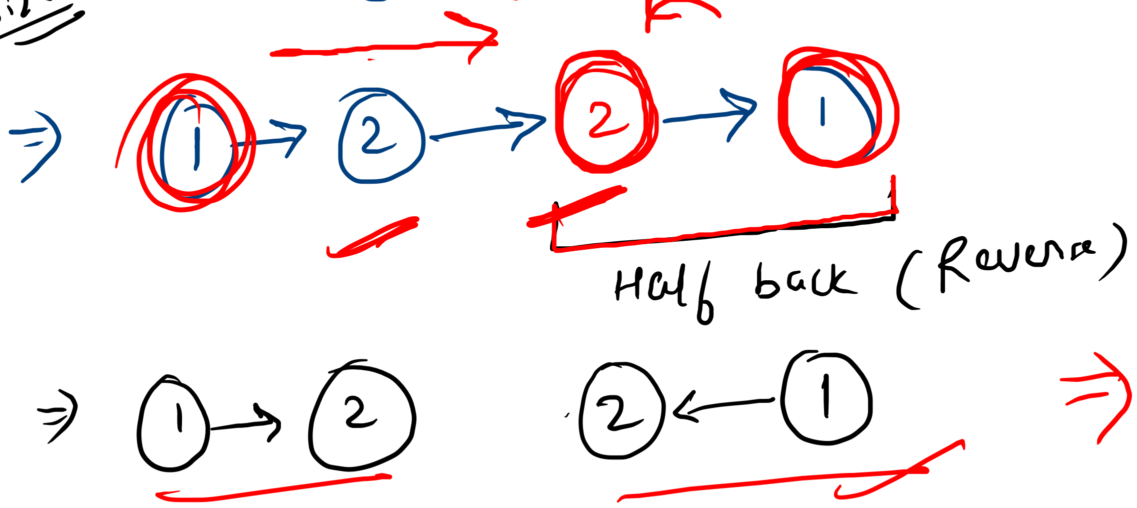
I/p: 1 → 2 → 2 → 1

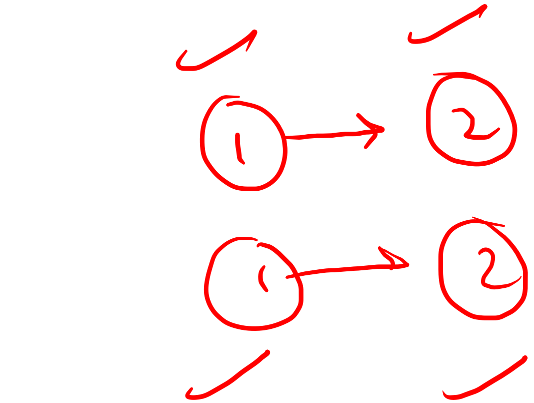
O/p:

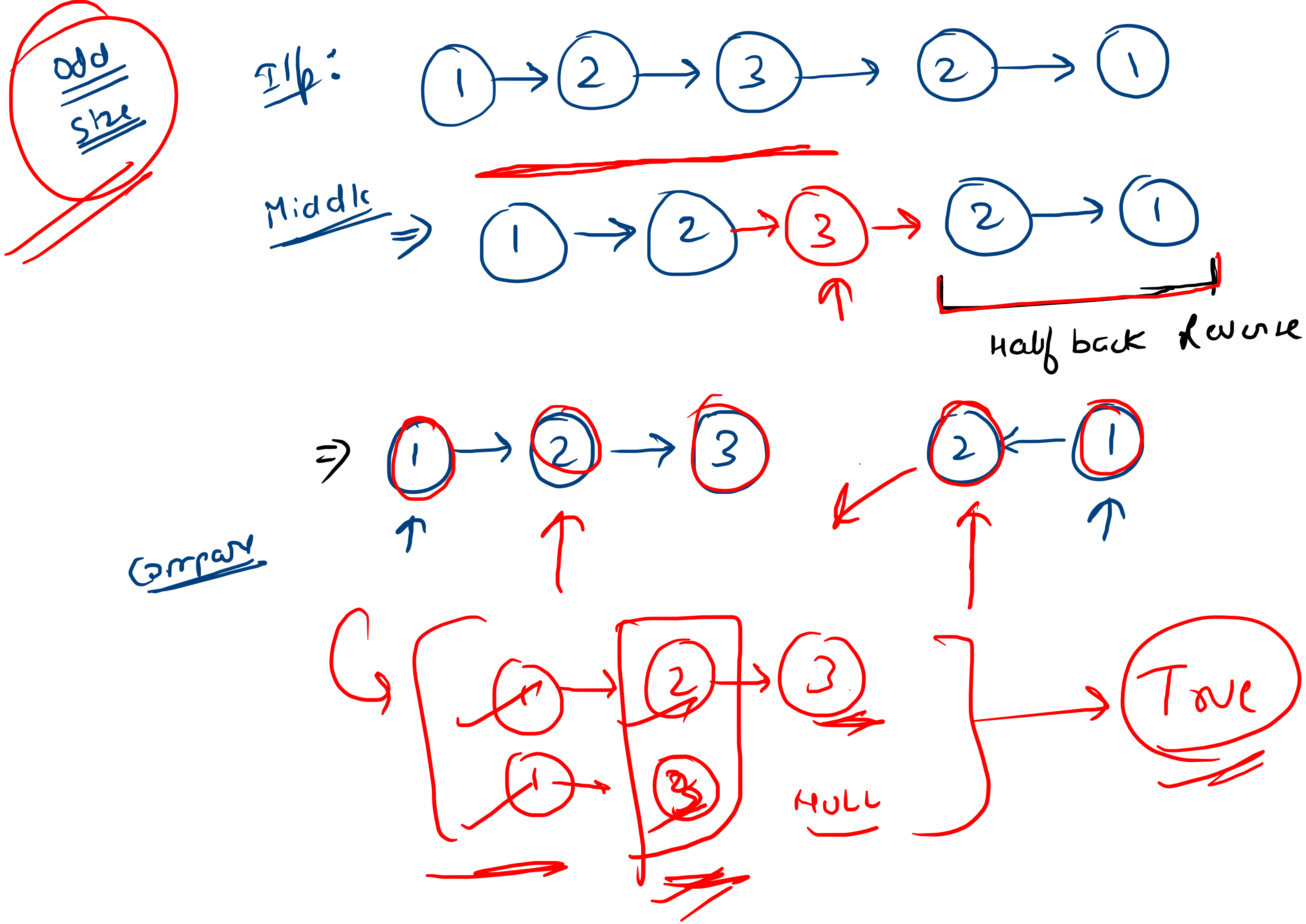
Idea:

1. Find middle of linked list
2. Reverse back Half of list
3. Compare the first Half and back Half

Even Size



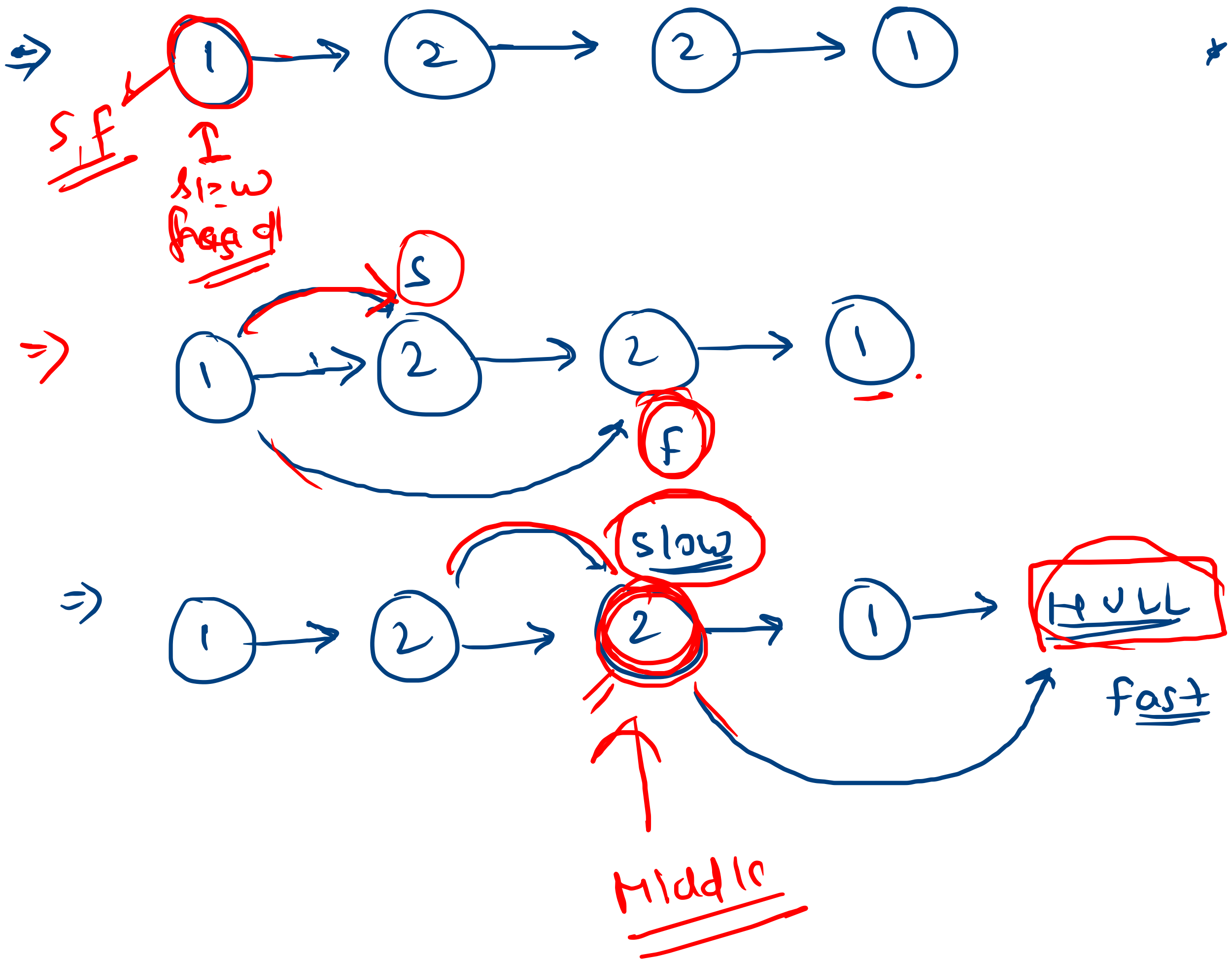



Odd Size
I/p:
1 → 2 → 3 → 2 → 1
Middle ⇒
1 → 2 → 3 → 2 → 1
Half back Reverse
⇒
1 → 2 → 3
2 ← 1
Compare
1 → 2 → 3
1 → 3
NULL
True

o> Steps to learn

- o> understand Logic
- o> Code that part of logic

Step 1 * find Middle in Linked List



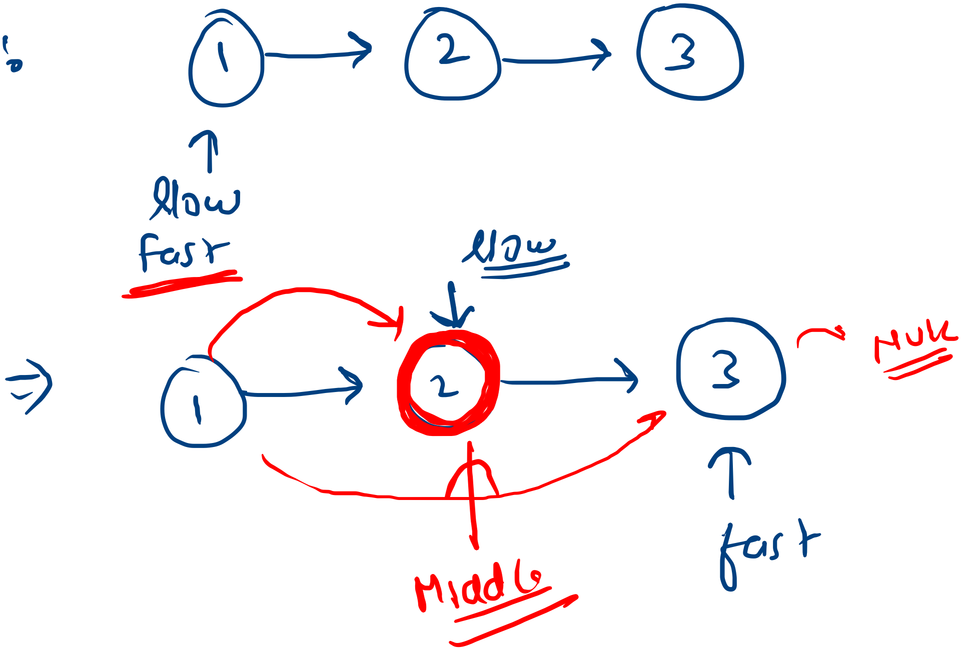
* slow = head
* fast = head → 2x speed

until

(fast != NULL &&
fast -> next != NULL)

odd

I/p:

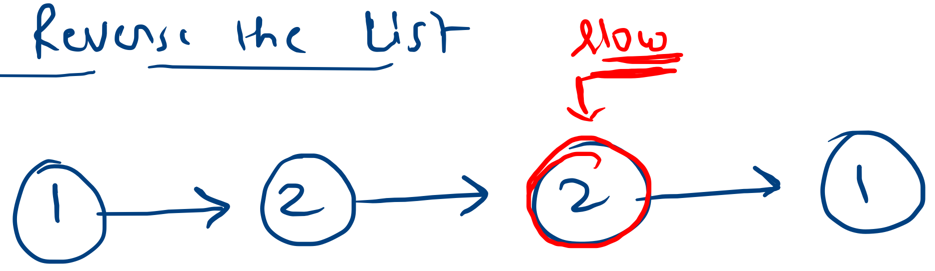


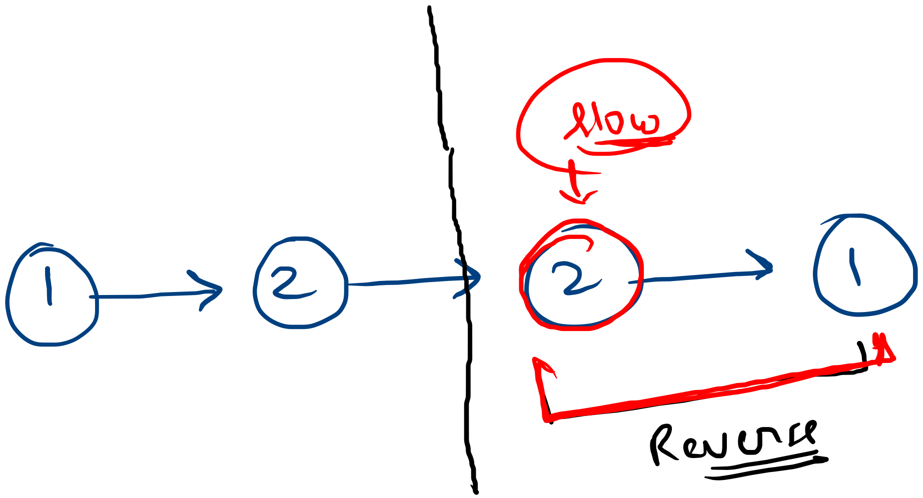
∴ slow is our middle of linked list

⇒ Code in Editor

Now Time to Reverse the List

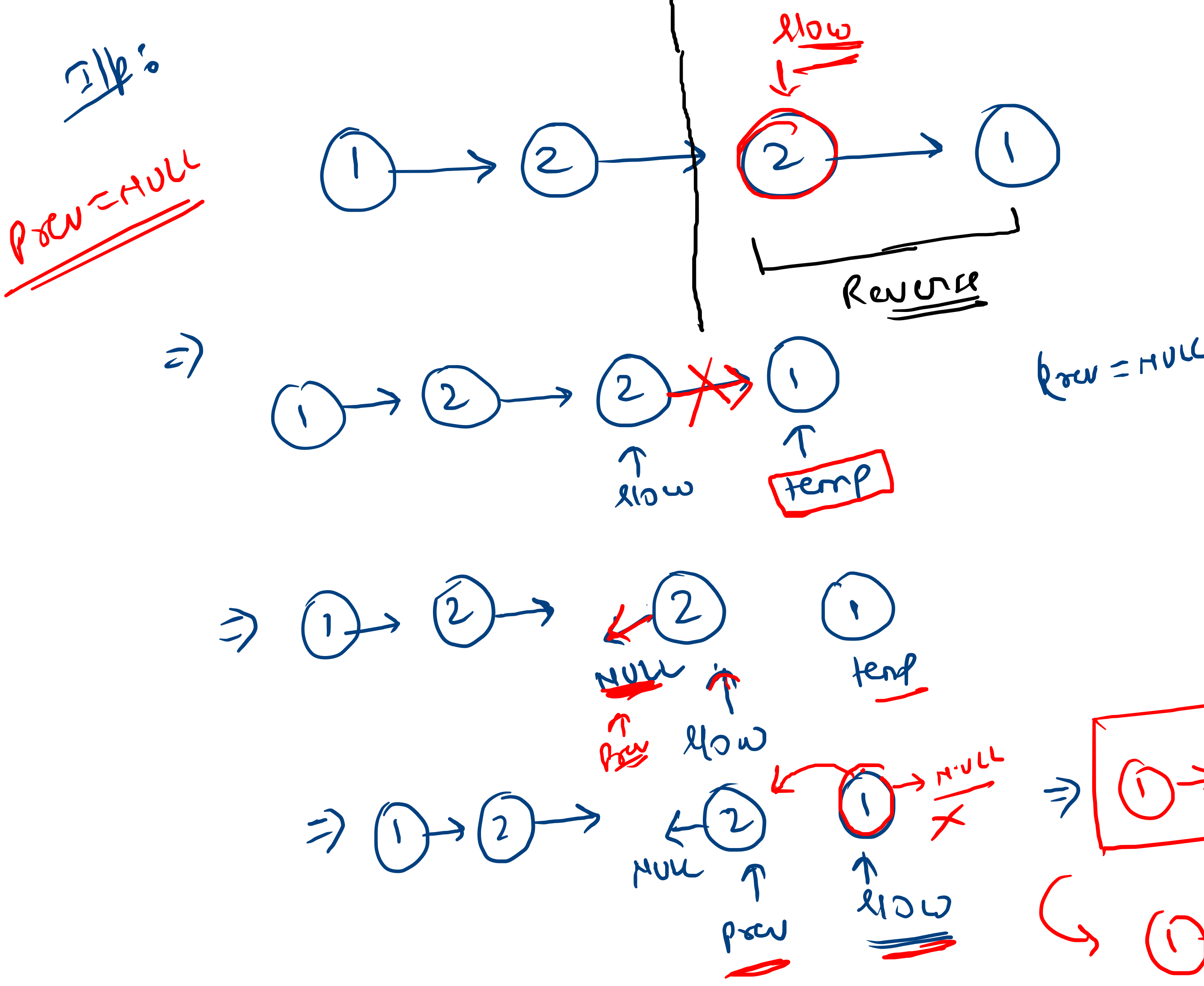
even I/p:



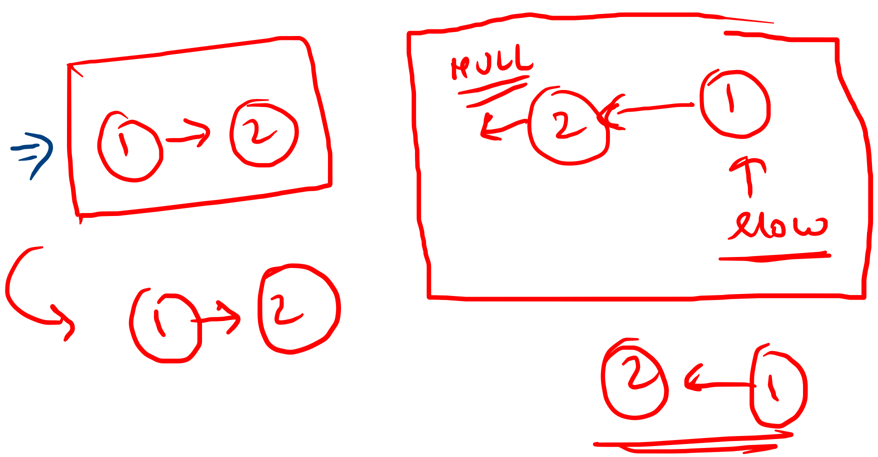


```
* prev = NULL

while( slow != NULL &&
       slow->next != NULL)
{
    * temp = slow->next;
    slow->next = prev;
    prev = slow;
    slow = temp;
}
```

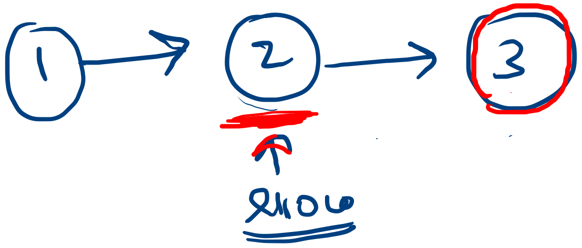


```
while( slow != NULL &&
        slow -> next != NULL)
{
   * temp = slow -> next;
     slow -> next = prev;
     prev = slow;
     slow = temp;
}
slow -> next = prev
```

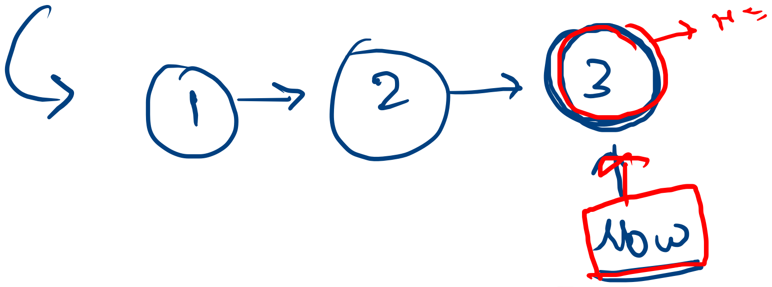


I/p: **odd**

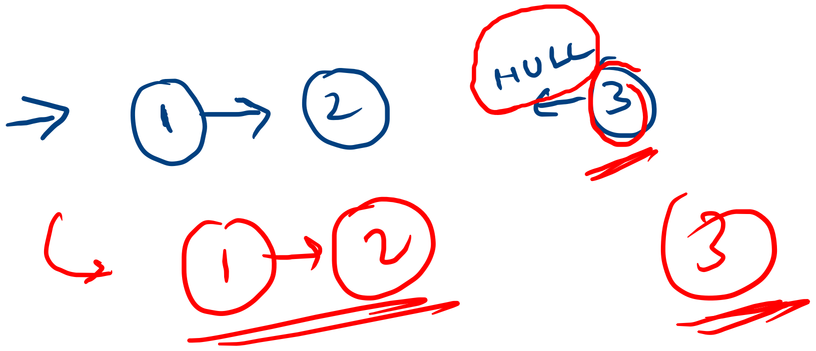
⇒



→ Shift slow in one node Ahead *



(prev = NULL)

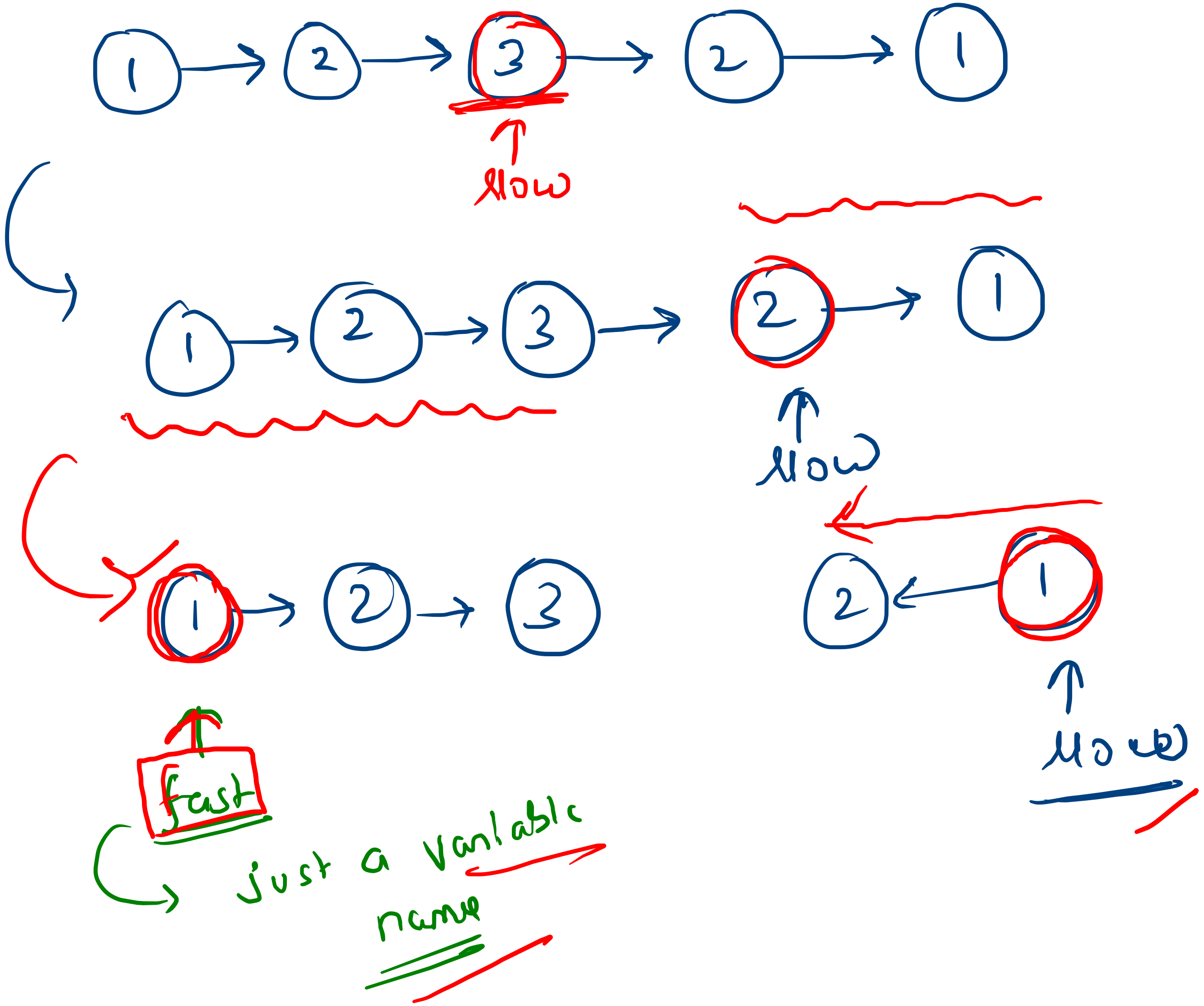


```
while( slow != NULL &&
       slow -> next != NULL)
{
   * temp = slow -> next;
     slow -> next = prev;
     prev = slow;
     slow = temp;
}
```

[slow -> next = prev]

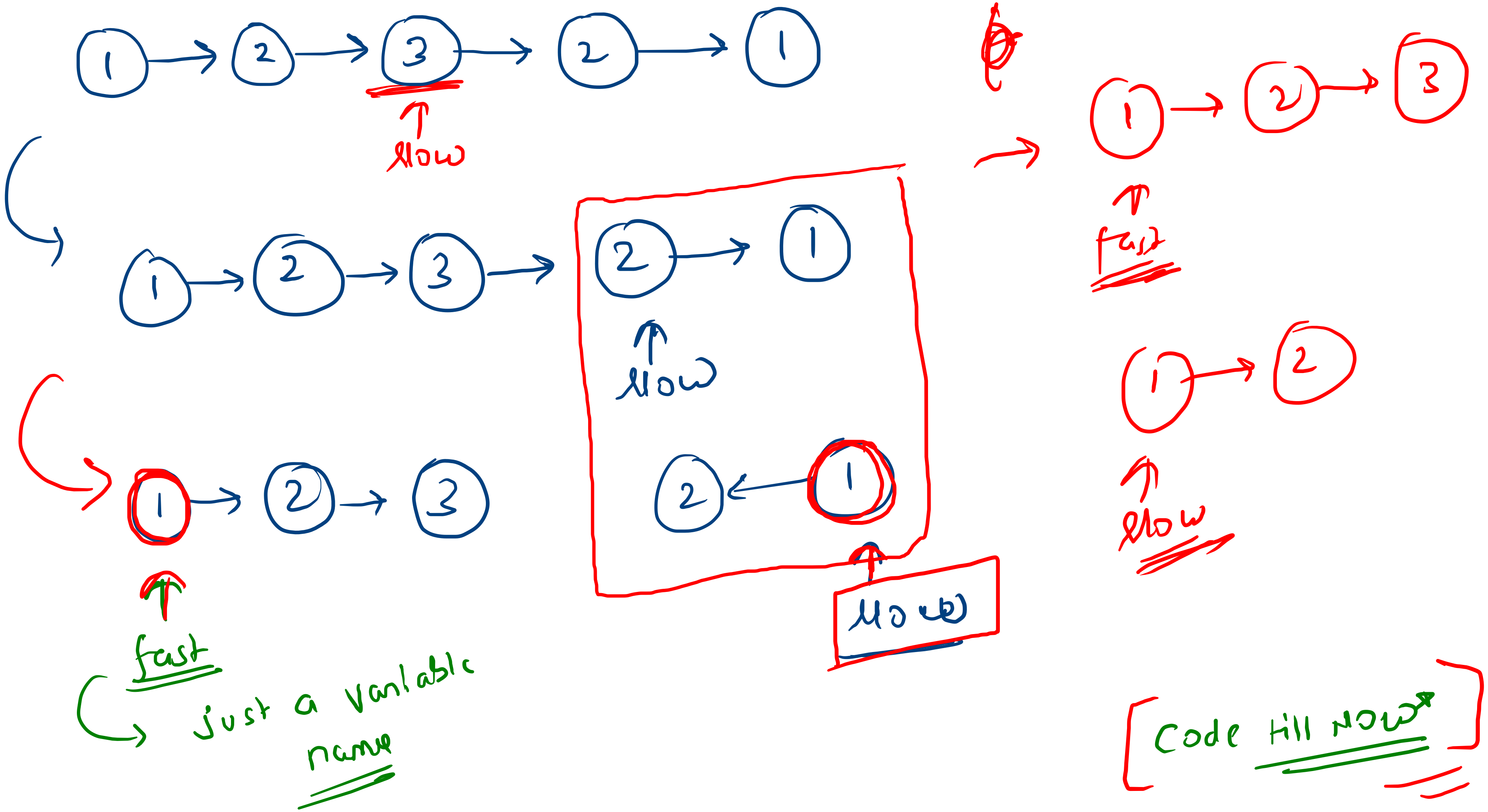
Overall idea

for odd no. of node Case



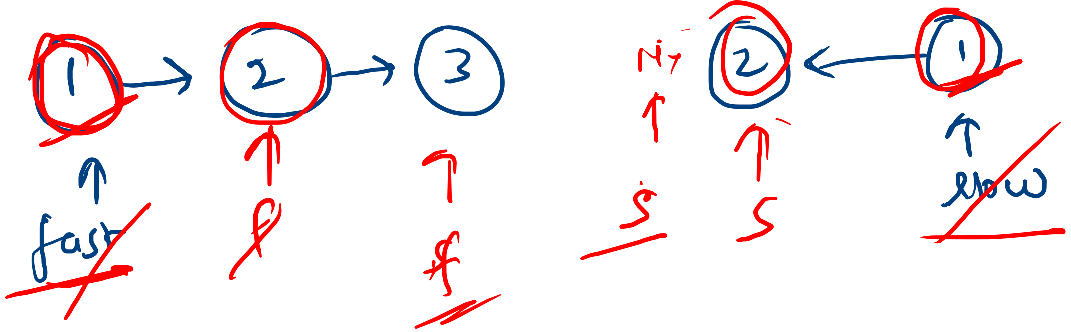
Overall idea

for odd no. of node case



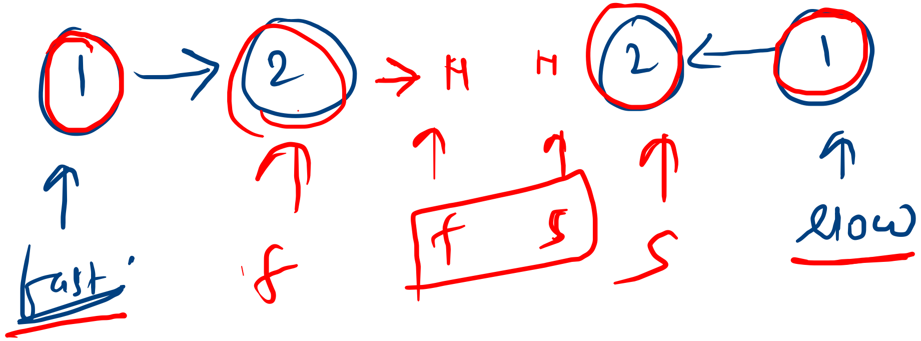
Code till now

I/P :



```
while( slow != NULL && fast != NULL)
{
    if( slow -> val    != fast -> val)
            return false;
    slow = slow -> next;
    fast = fast -> next;
}
return True;
```

I/p:



return true!

Let's code it