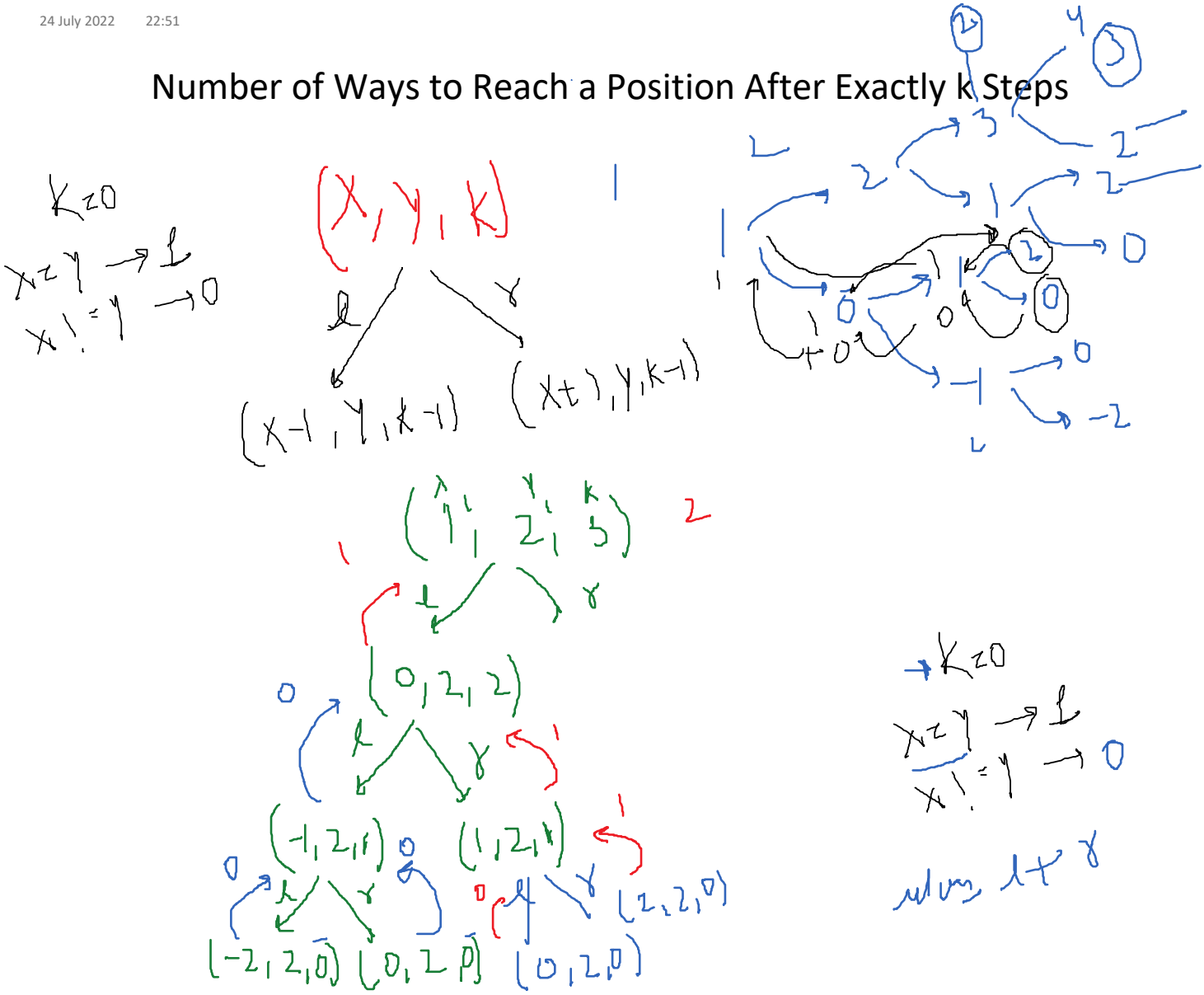
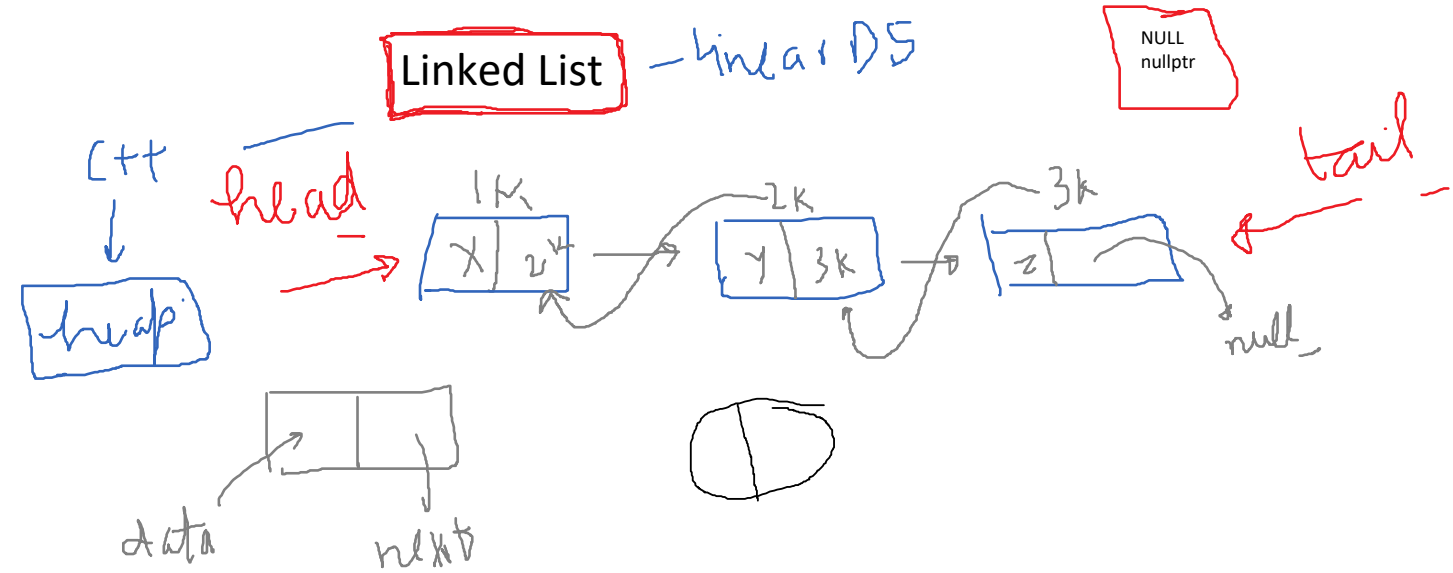
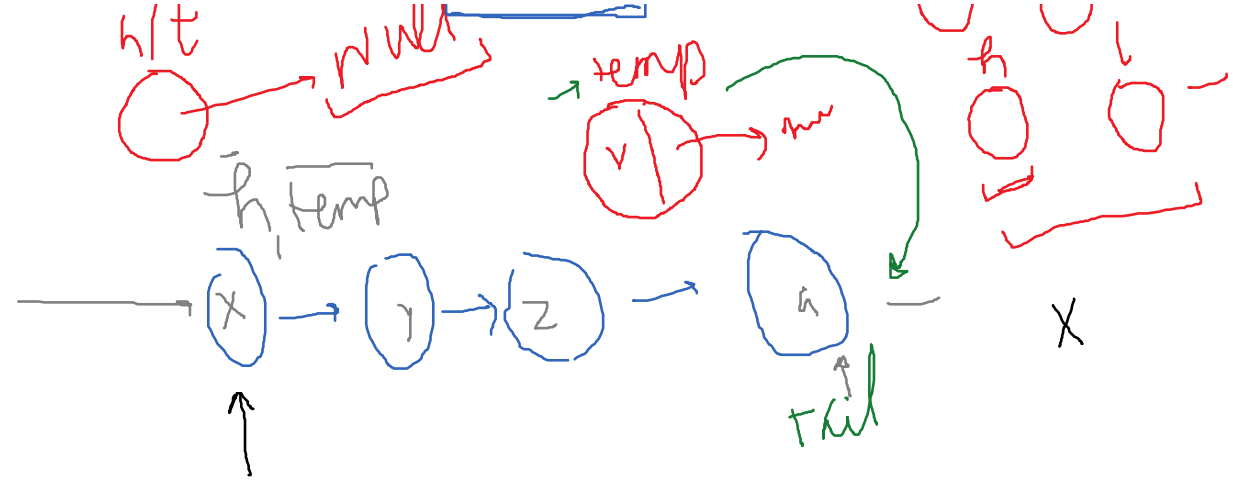
Linked List 1

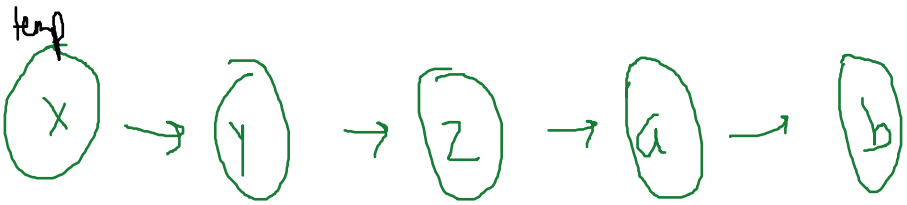
24 July 2022 22:51

Number of Ways to Reach a Position After Exactly k Steps

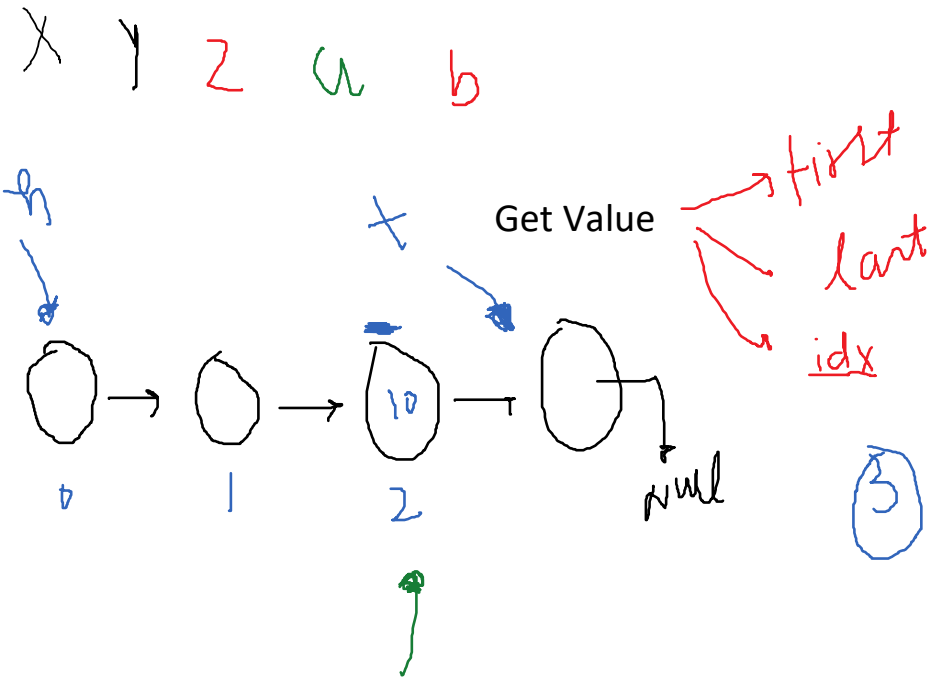




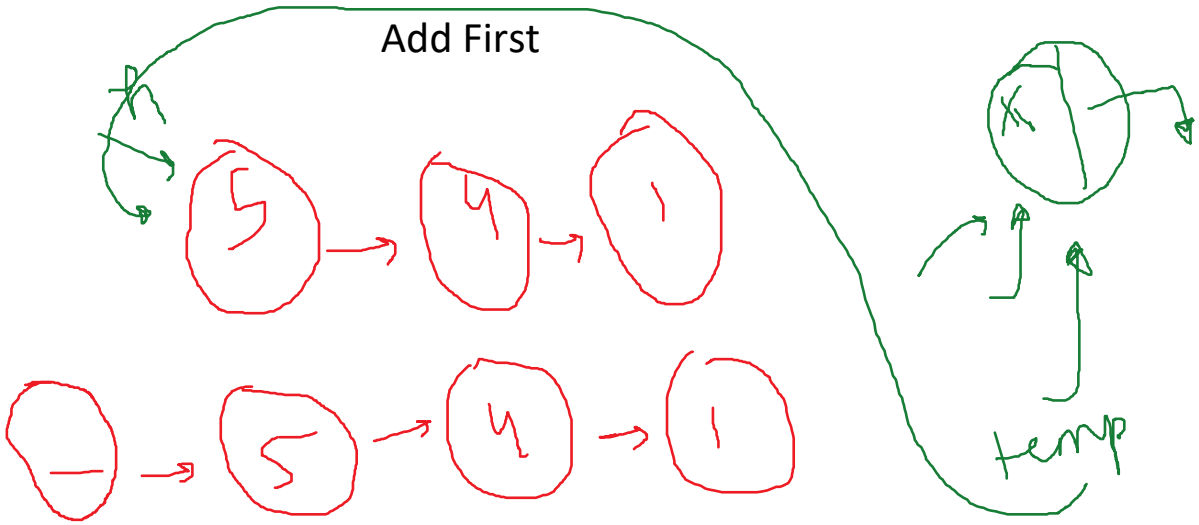


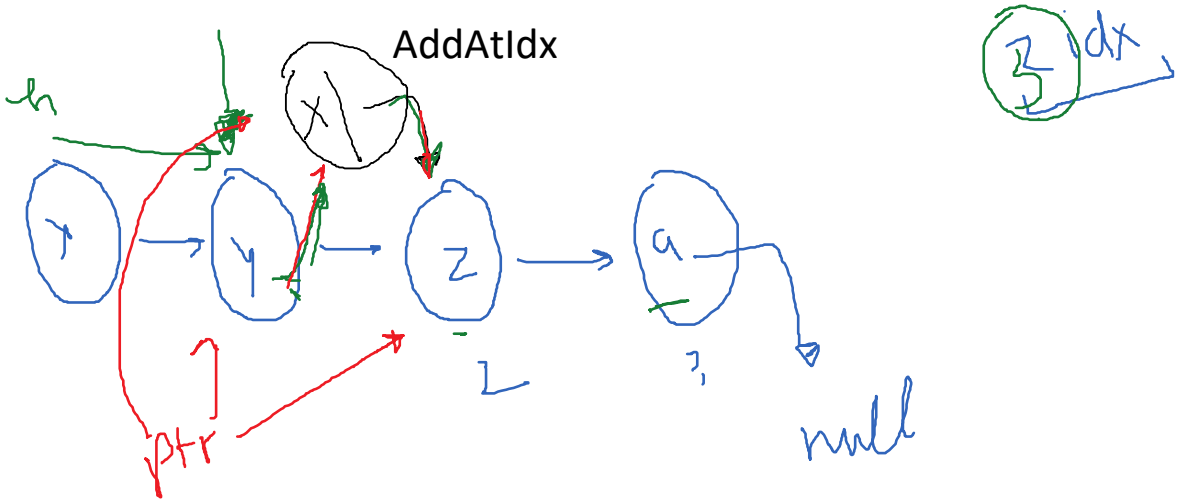


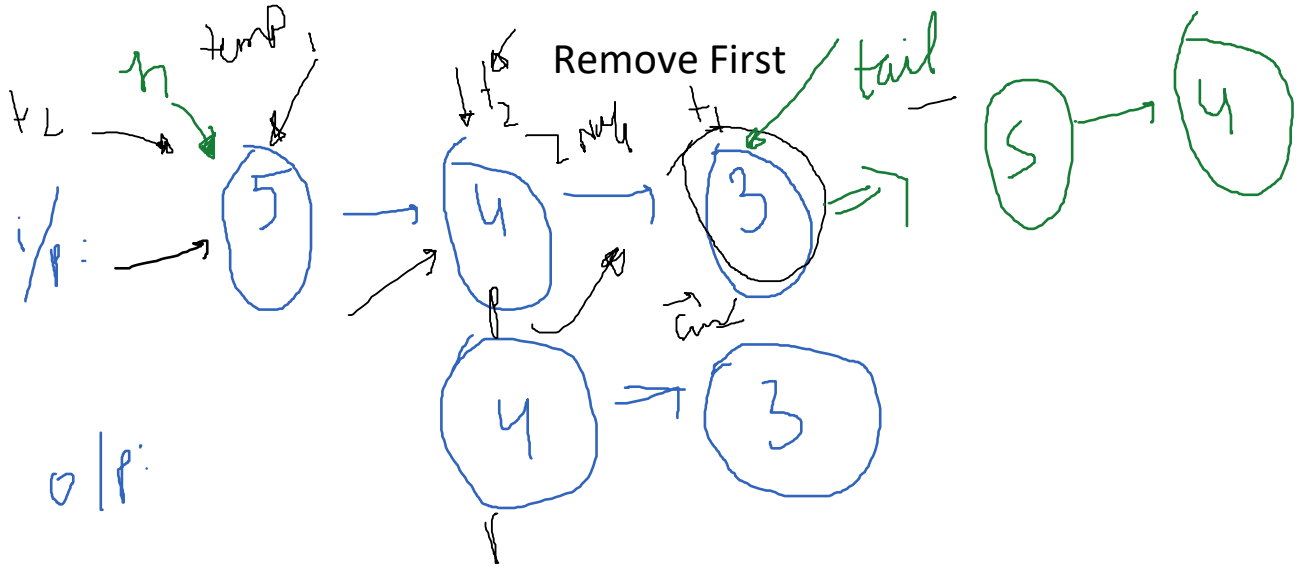

```
void display()
{
    cout<<"Our Linked List is : ";
    Node * temp = head;
    while(temp == NULL)
    {
        cout<< temp->data<< " ";
        temp = temp->next;
    }
    cout<< endl;
}
```




```
void getAtIdx(int idx)
{
    int cnt = 1 ;
    Node *temp = head;
    while(cnt != idx)
    {
        temp = temp->next;
        cnt++;
    }
    cout<<"our Idx value is : " <<  temp->data << endl;
}
```


Add First
temp

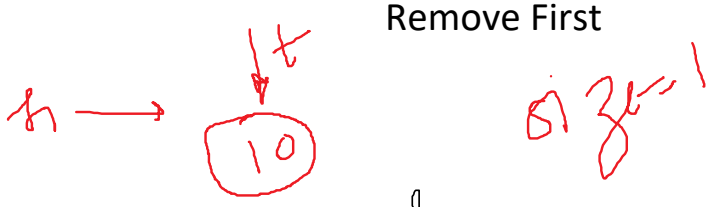

AddAtIdx
idx
ptr
null


Remove First
temp
tail
i/p:
o/p:

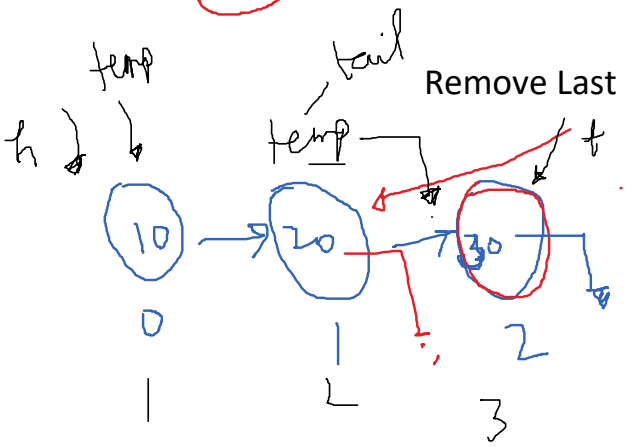
Linked List 2

12 September 2022 14:17

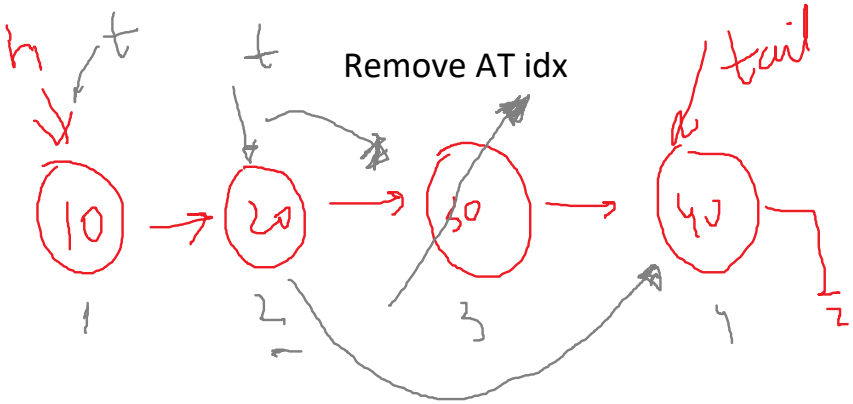
Remove First



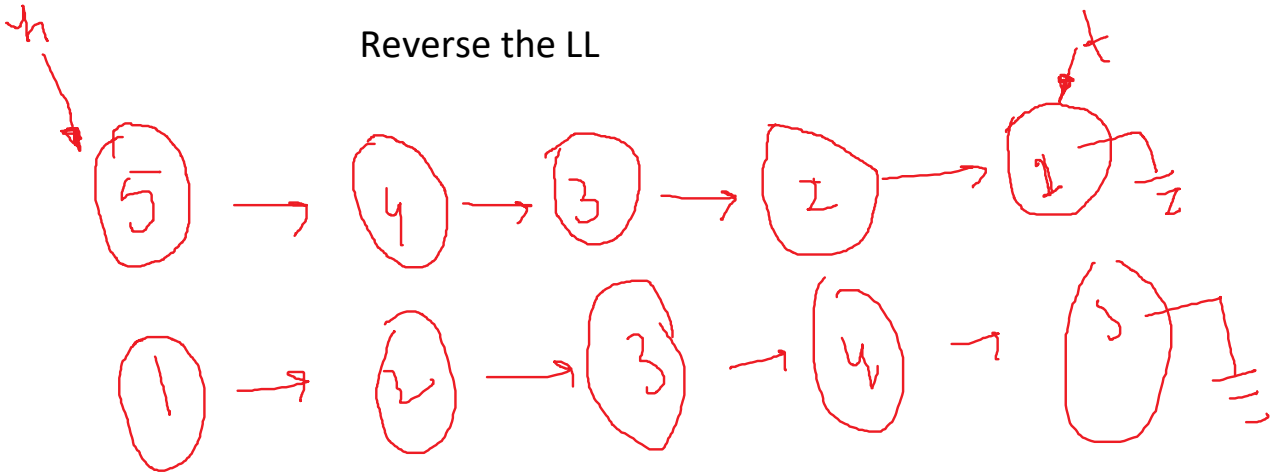
Remove Last

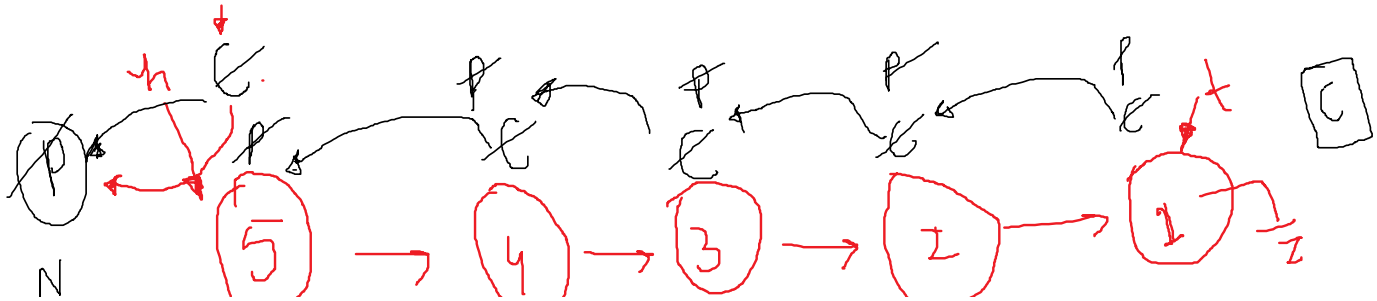


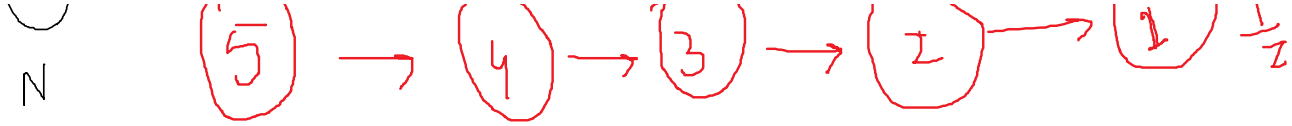
Remove AT idx

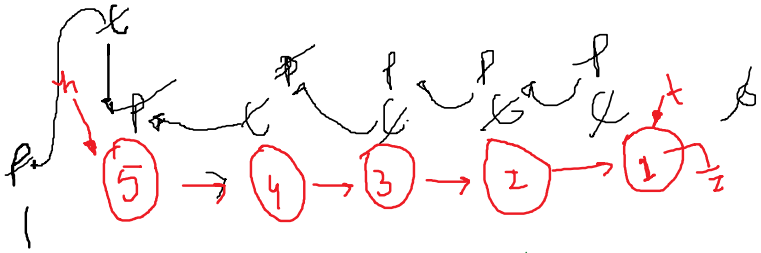


Reverse the LL

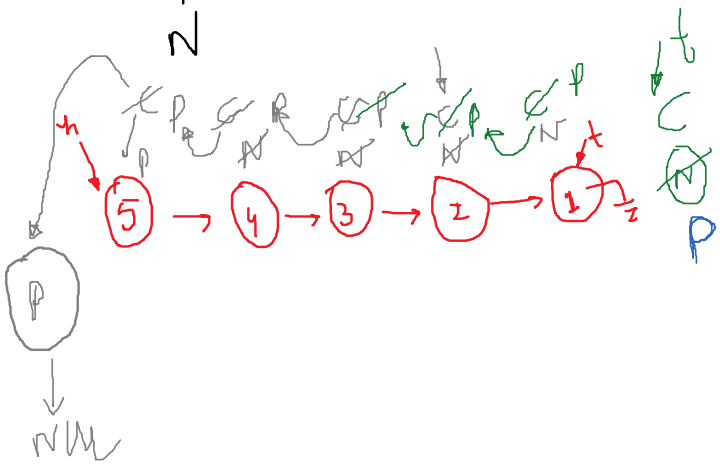








```
void reversePtr()
{
    Node * prev = NULL;
    Node * curr = head;
    while(curr != NULL)
    {
        curr->next = prev;

        prev = prev->next ;
        curr= curr->next;
    }
}
```

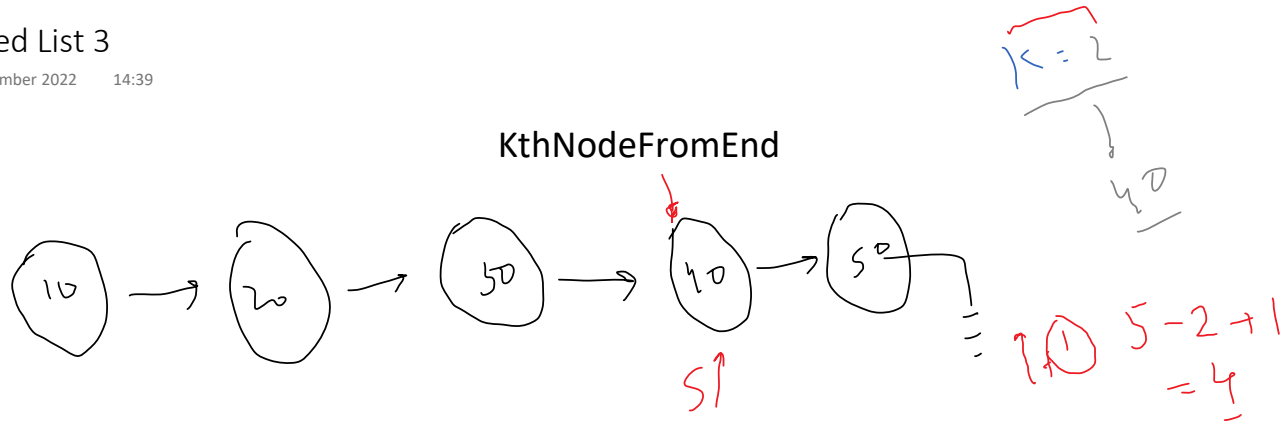


```
void reversePtr()
{
    Node * prev = NULL;
    Node * curr = head;
    while(curr != NULL)
    {
        Node * nextnode  = curr->next;
        curr->next = prev;
        prev = curr;
        curr = nextnode;


    }
}
```

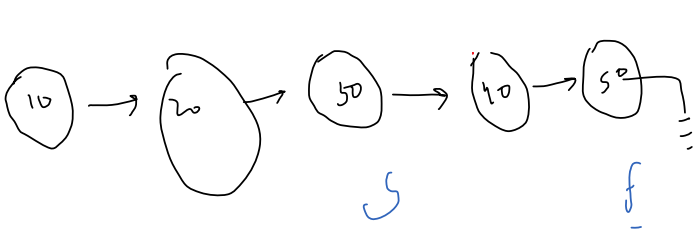
Linked List 3

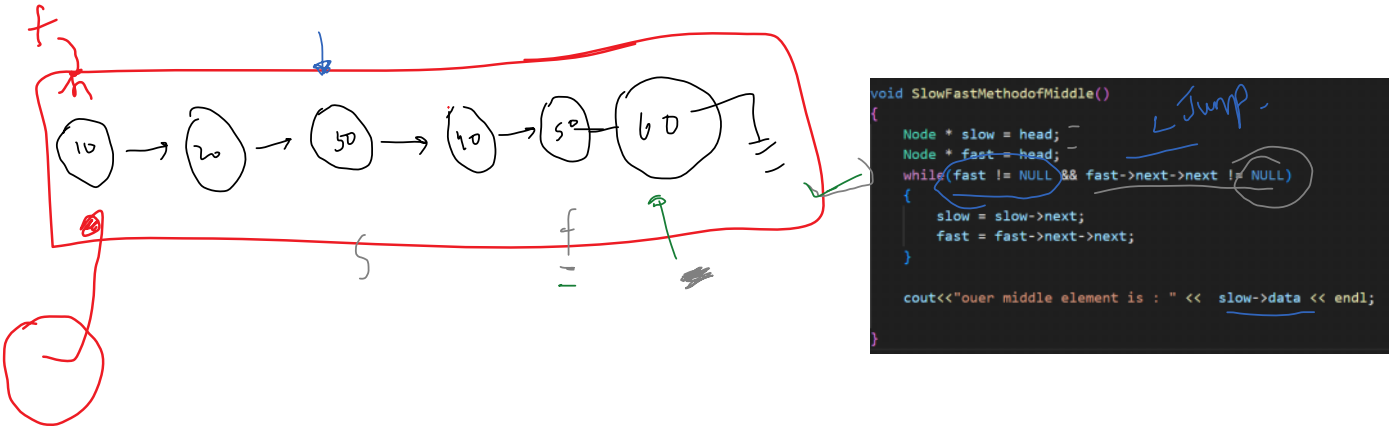
13 September 2022 14:39

KthNodeFromEnd



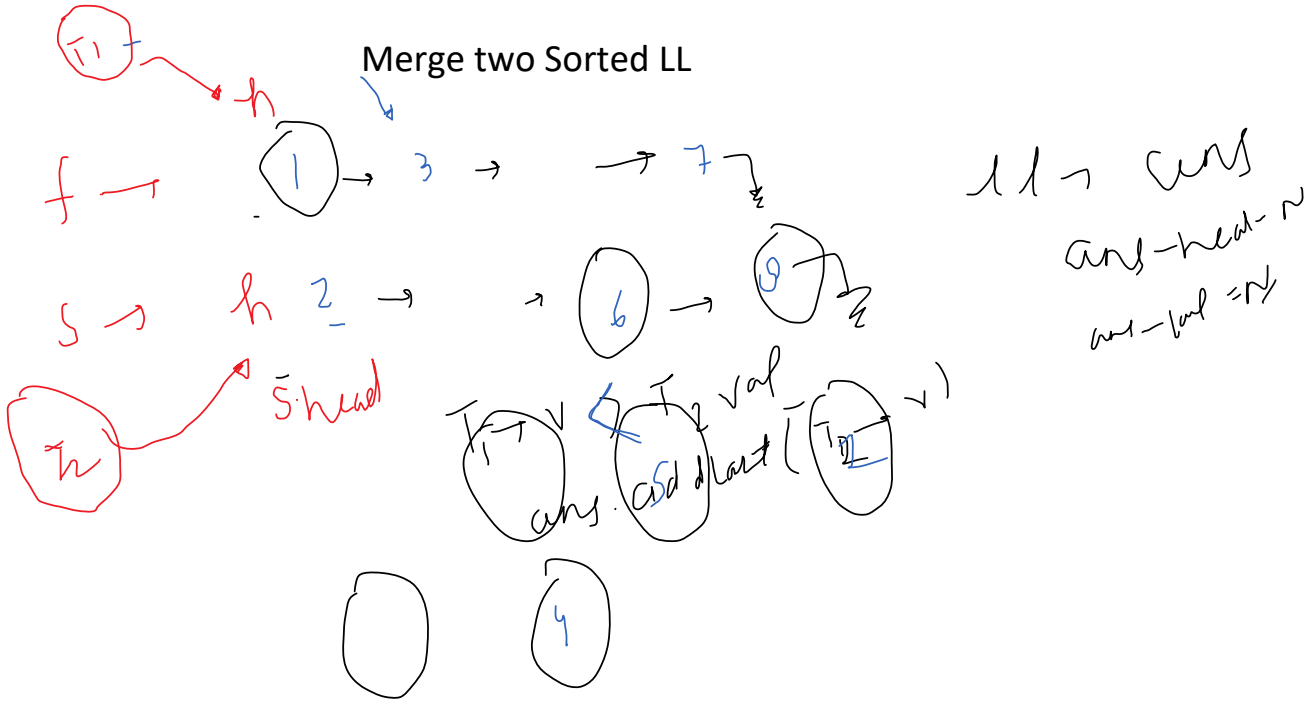
Middle of Linked List





```
void SlowFastMethodofMiddle()
{
    Node * slow = head;
    Node * fast = head;
    while(fast != NULL && fast->next->next != NULL)
    {
        slow = slow->next;
        fast = fast->next->next;
    }

    cout<<"ouer middle element is : " << slow->data << endl;
}
```

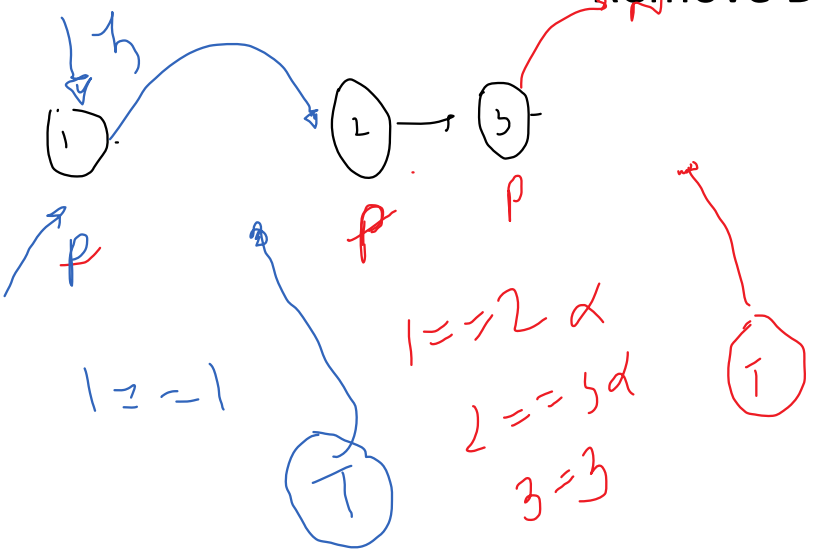
Merge two Sorted LL



Linked List 4

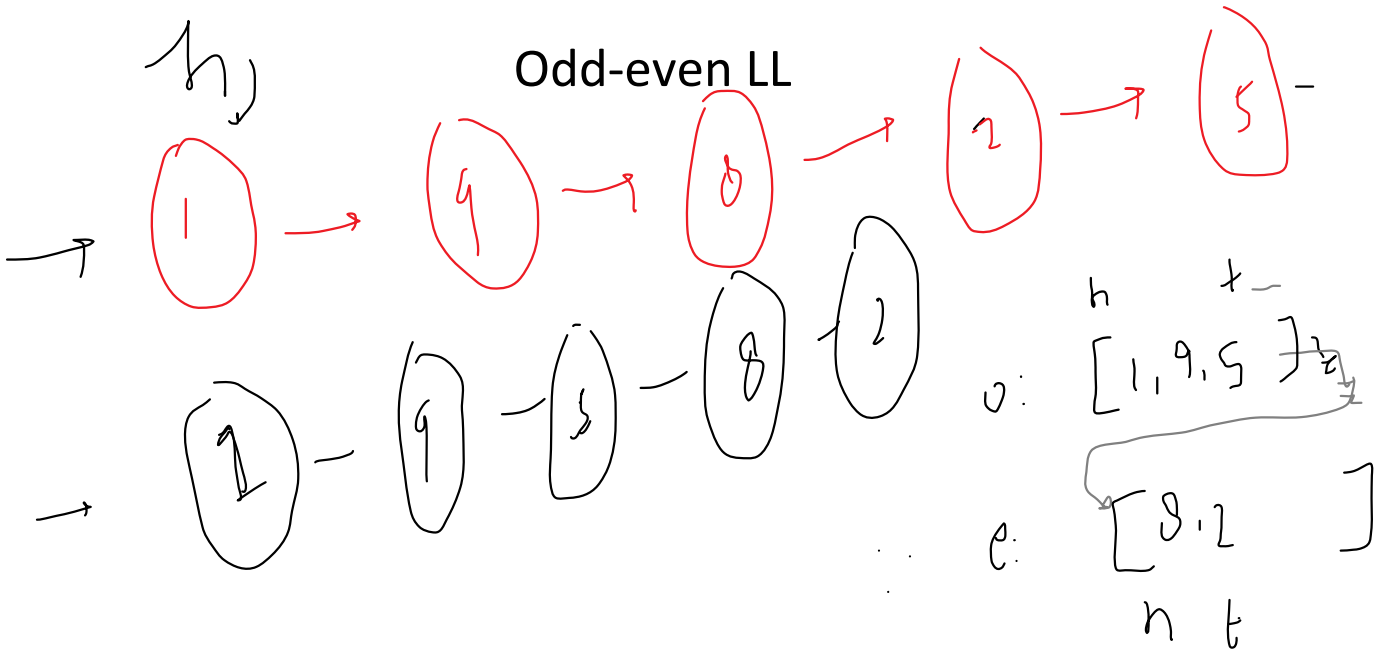
14 September 2022 15:12

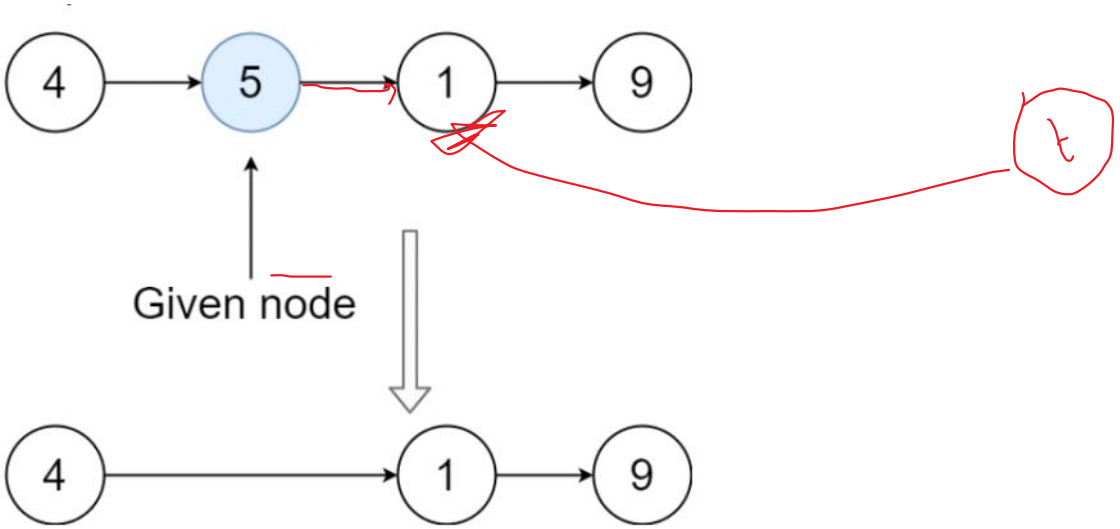
Remove Duplicate



```
ListNode * prev = head;
while(prev->next != NULL)
{
    if(prev->val == prev->next->val)
    {
        ListNode * temp = prev->next;
        prev->next = prev->next->next;
        delete temp;
    }
    else{
        prev = prev->next;
    }
}
return head;
```

Odd-even LL



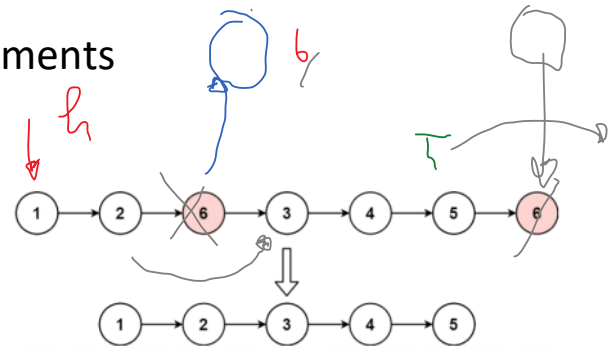


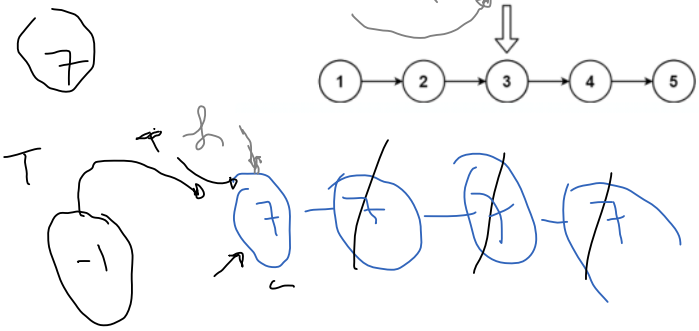
Linked List 5

16 September 2022 14:32

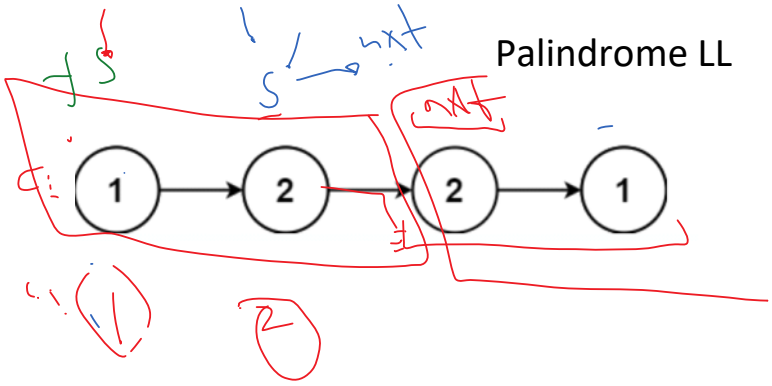
Remove Linked List Elements

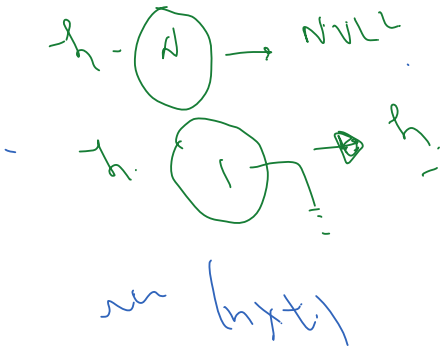
```
ListNode* removeElements(ListNode* head, int val)
{
    if(head == NULL) return head;
    ListNode * dummy = new ListNode(-1);
    dummy->next = head;
    ListNode * temp = head;
    while(temp != NULL && temp->next != NULL)
    {
        if(temp->next->val == val)
        {
            ListNode * nextNode = temp->next;
            temp->next = temp->next->next;
            delete(nextNode);

        }
        else{
            temp= temp->next;
        }
    }
    return
```



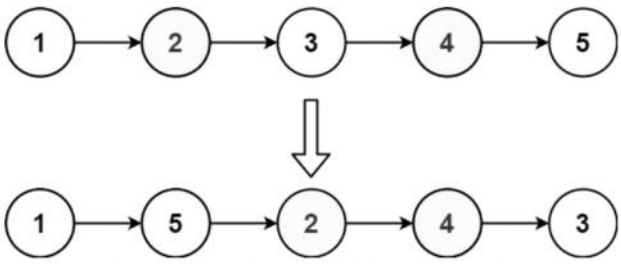


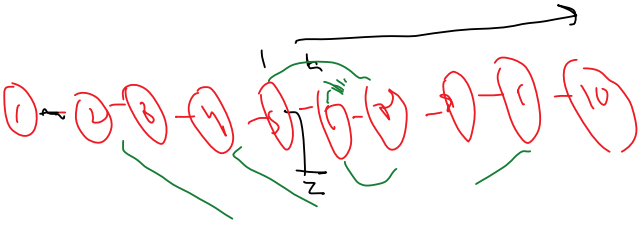
Palindrome LL





Fold Linked List





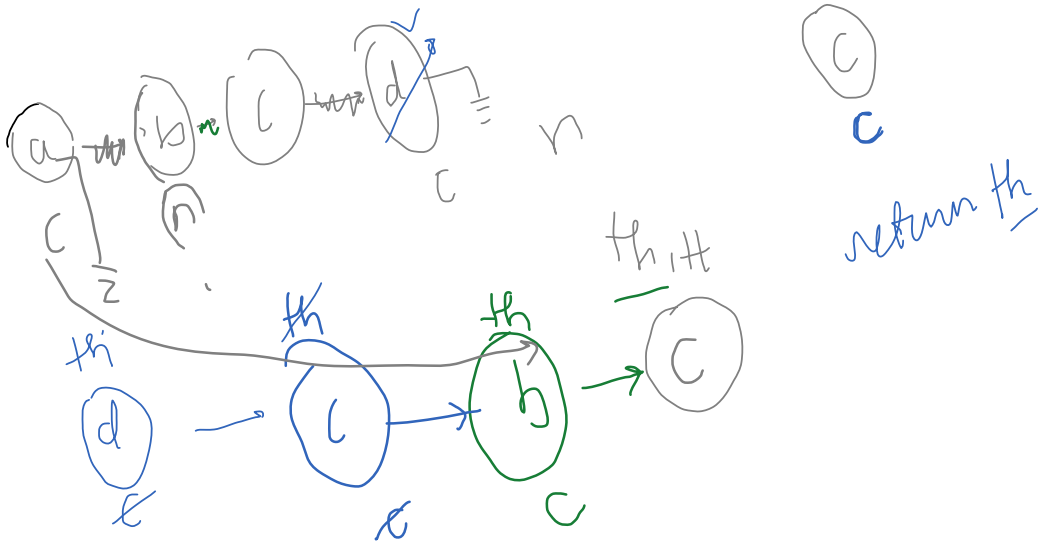
1->2->3->4->5

10->9->8->7->6

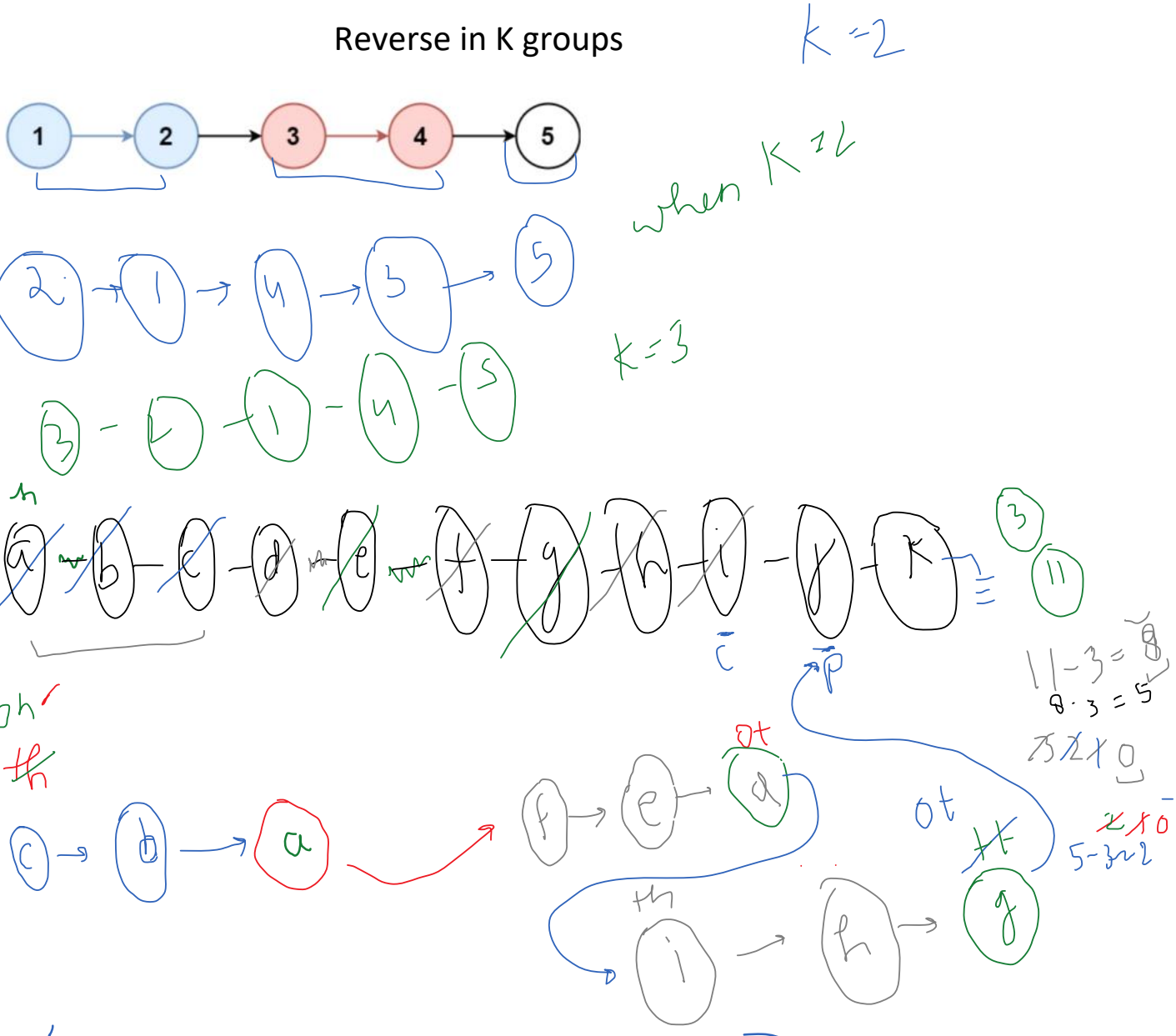
Linked List 6

18 September 2022 12:22

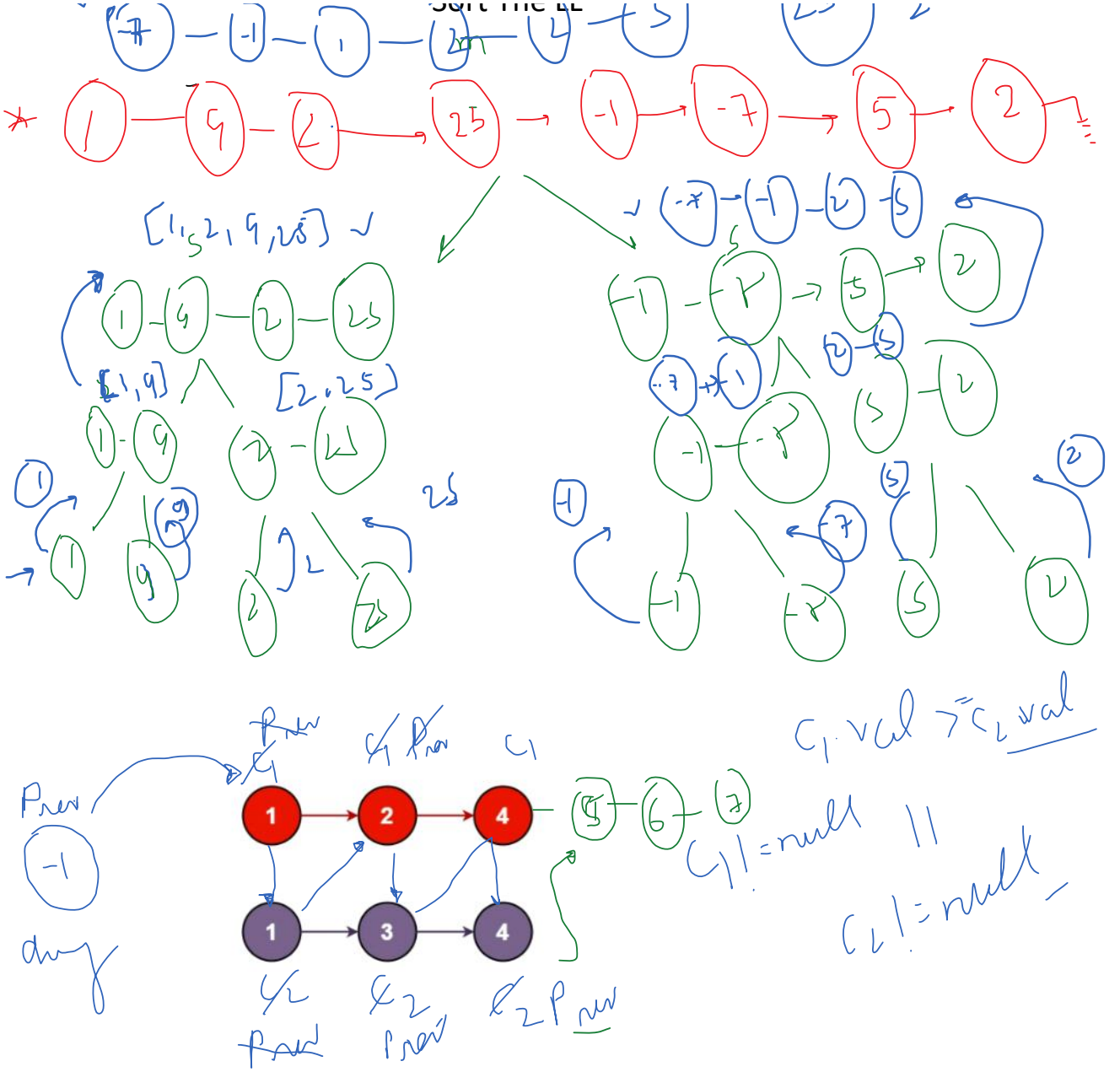
REVERSE A LL



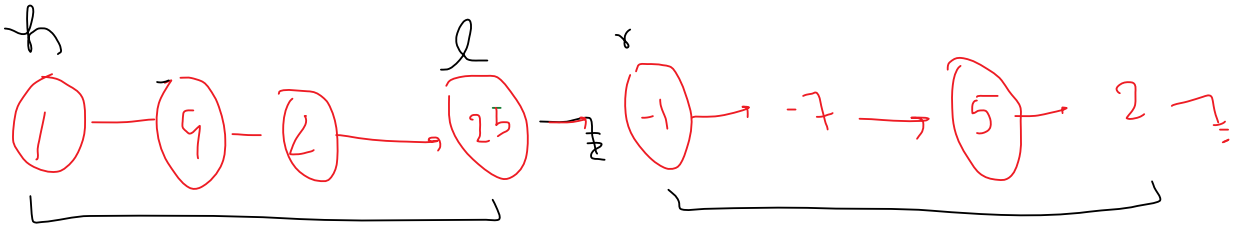
Reverse in K groups



Sort The LL



1
2
4
5
6
7
1
3
4



$(call)^{h} + (prev * h) + (pos * h)$

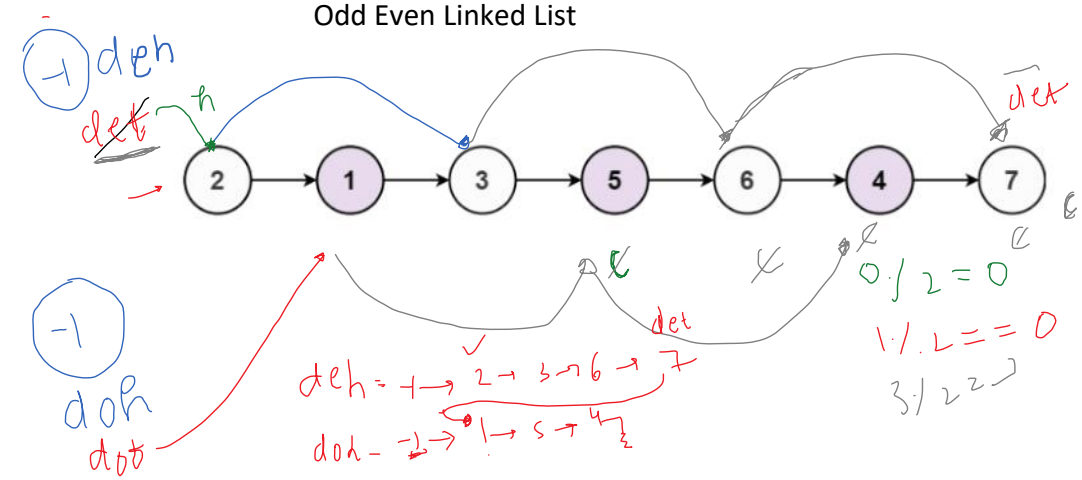
$+ \quad (2)^{h} + (K * 4) + (O(n) * 4)$

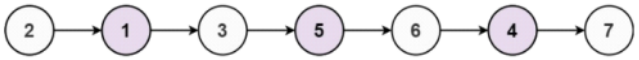
$=$

Linked List 7

19 September 2022 19:43

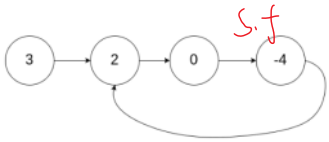
Odd Even Linked List





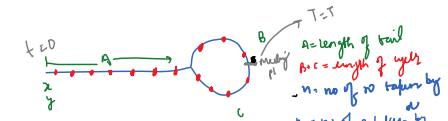

```
ListNode* oddEvenList(ListNode* head)
{
    ListNode * evenhead = new ListNode (-1);
    ListNode * oddhead = new ListNode(-1);
    ListNode * eventail = evenhead;
    ListNode * oddtail = oddhead;
    ListNode * curr = head;
    int  cnt = 0 ;
    while(curr != NULL)
    {
        if(cnt % 2 == 0)
        {
            eventail->next = curr;
            eventail = eventail->next;
        }
        else{
            oddtail->next = curr;
            oddtail = oddtail->next;
        }
        curr = curr->next;
        cnt++;
    }
    eventail->next = oddhead->next;
    oddtail->next = NULL;
    return evenhead->next;
}
```

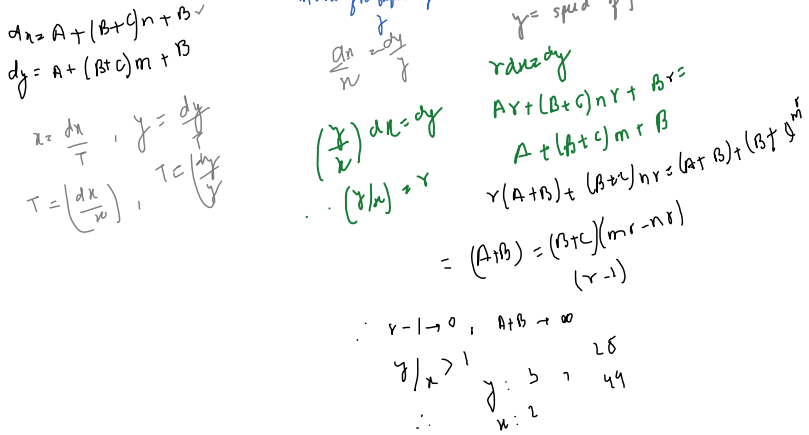
Cycle detection



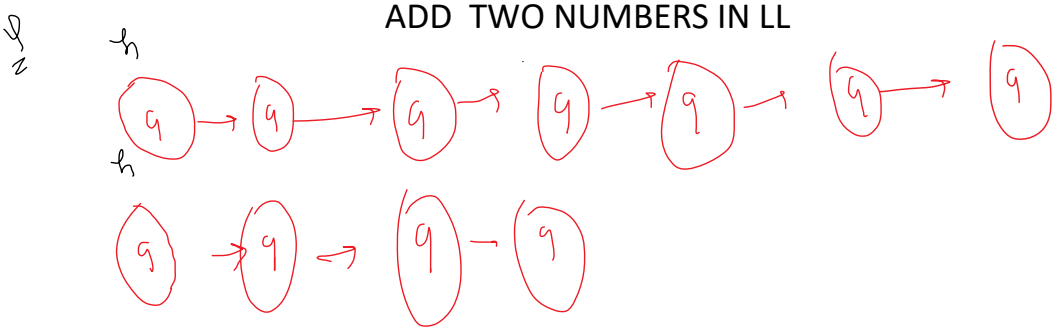
Floyd cycle det

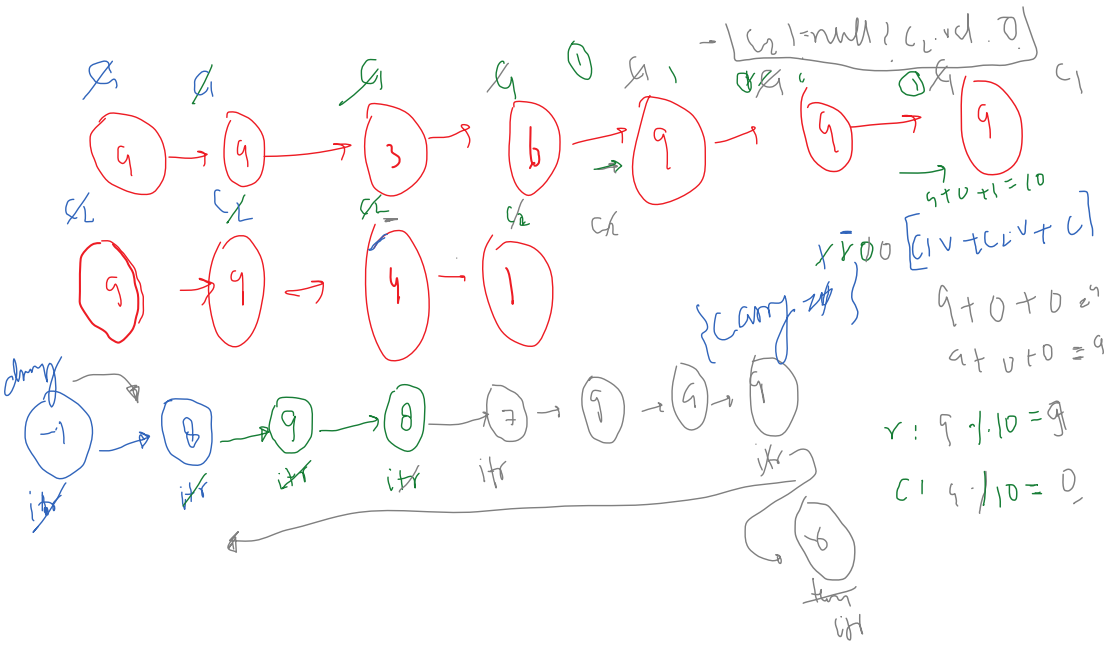
Read - for trivial algo





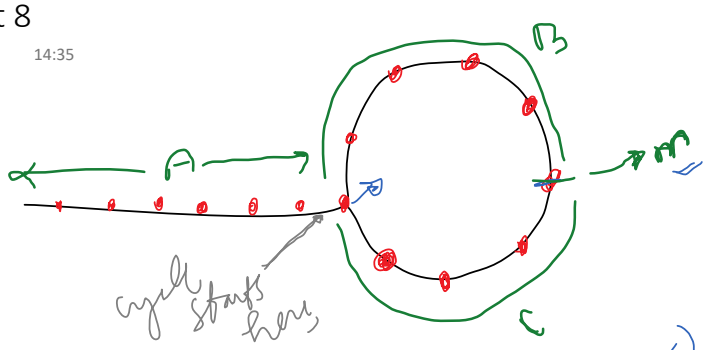
ADD TWO NUMBERS IN LL

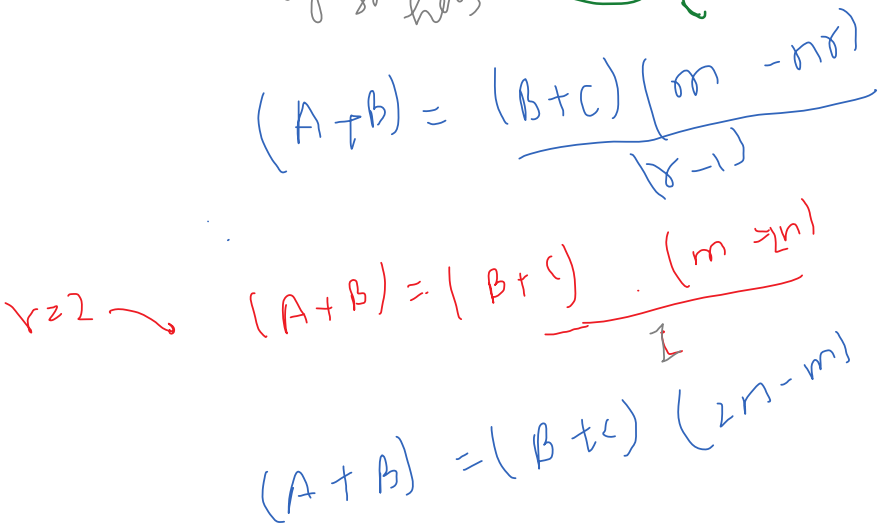


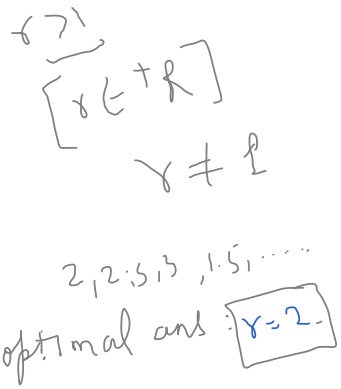


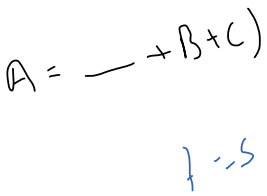
Linked List 8

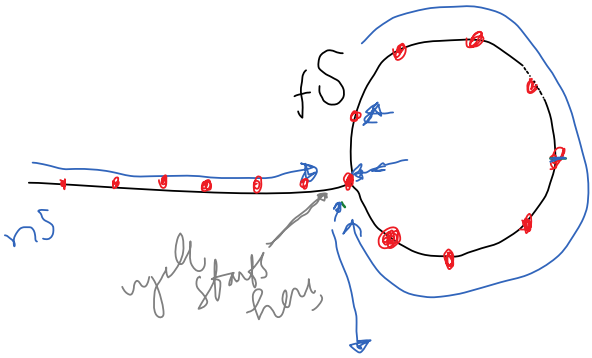
21 September 2022 14:35



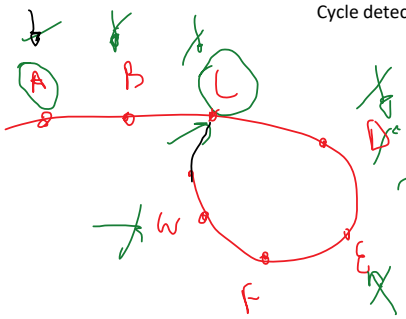




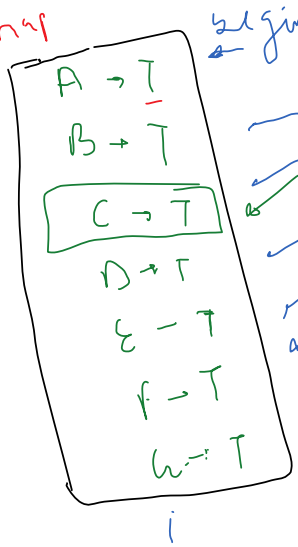


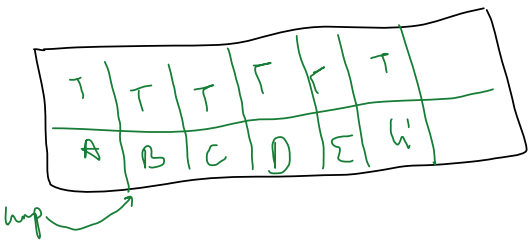


Cycle detection Method 2



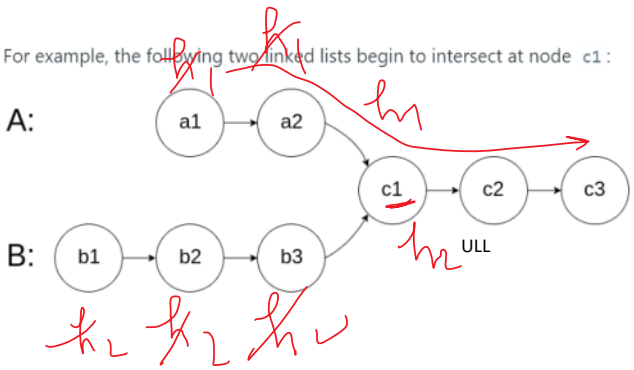

```
if(ump.find(curr) != ump.end()) return true;
else{
    ump[curr] = true;
}
curr = curr -> next;
```



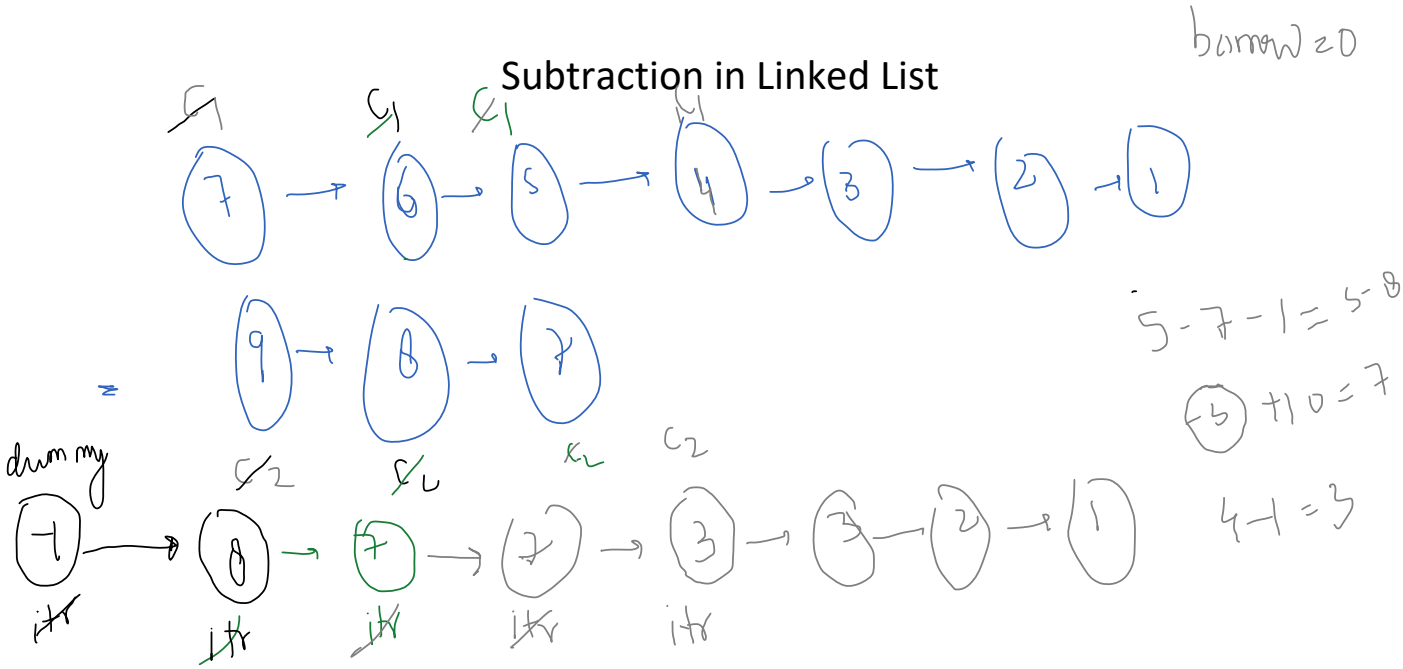


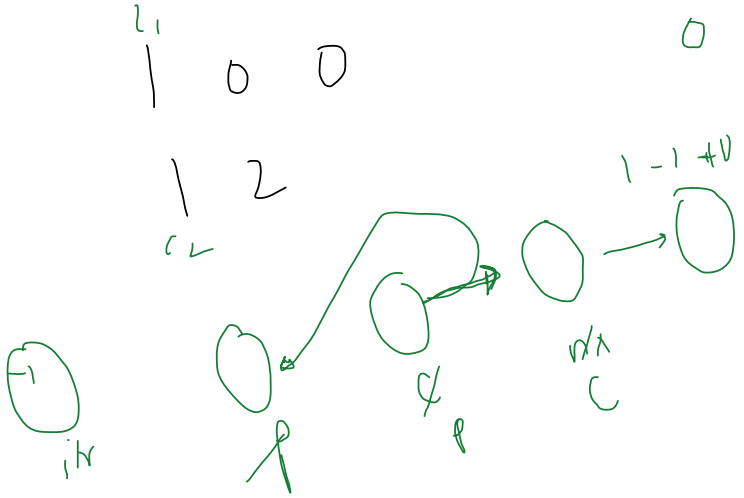
Intersection of Two LL

For example, the following two linked lists begin to intersect at node `c1`:



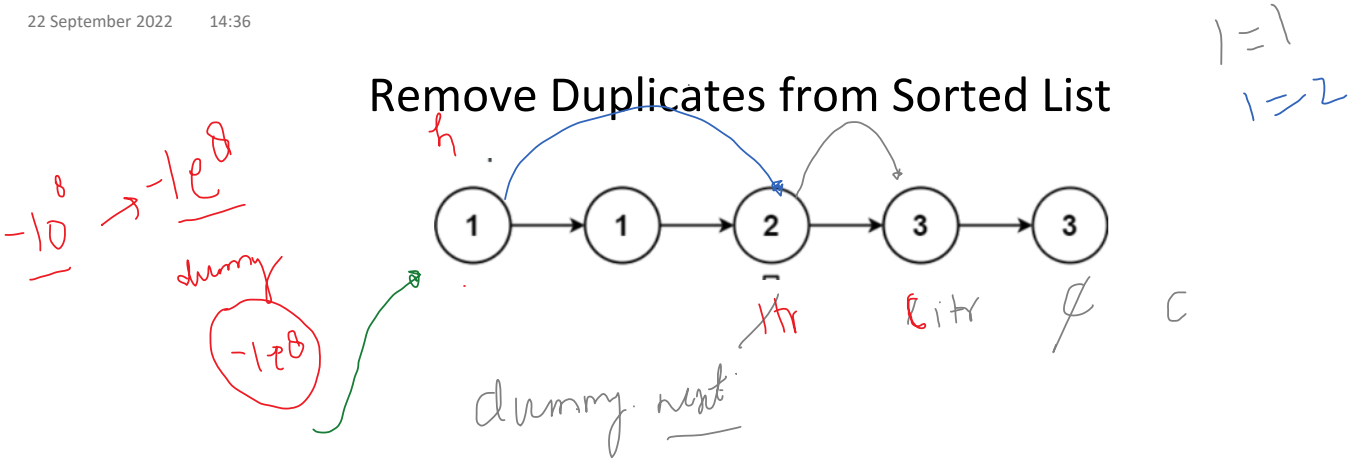
Subtraction in Linked List

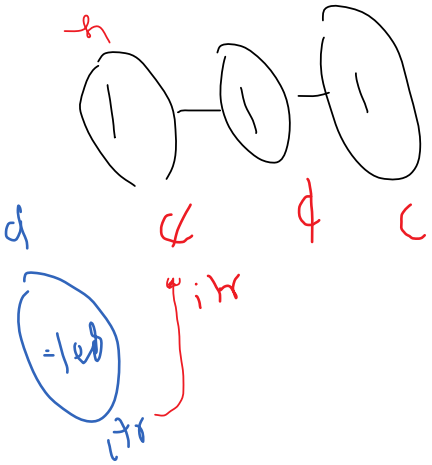




```
Node* subLinkedList(Node* l1, Node* l2)
{
   Node * c1 = l1;
   Node * c2 = l2;
   Node * dummy = new Node(-1);
   Node * itr = dummy;
   int borrow = 0 ;
   while(c1 != NULL || c2 != NULL || borrow != 0)
   {
       int val1 = (c1 != NULL) ? c1 ->data : 0 ;
       int val2 = (c2 != NULL) ? c2 ->data : 0 ;
       int sum = val1 - val2  + borrow;
       cout<< sum << endl;
       if(sum < 0)
       {
           sum += 10;
           borrow = -1;
       }
       else{
           borrow = 0;
           sum = sum;
       }
       Node * temp = new Node(sum);
       itr->next = temp;
       itr = itr->next;
       if(c1 != NULL)
            c1 = c1->next;
       if(c2 != NULL )
            c2 =  c2-> next;

   }
   return dummy-> next;

}
```

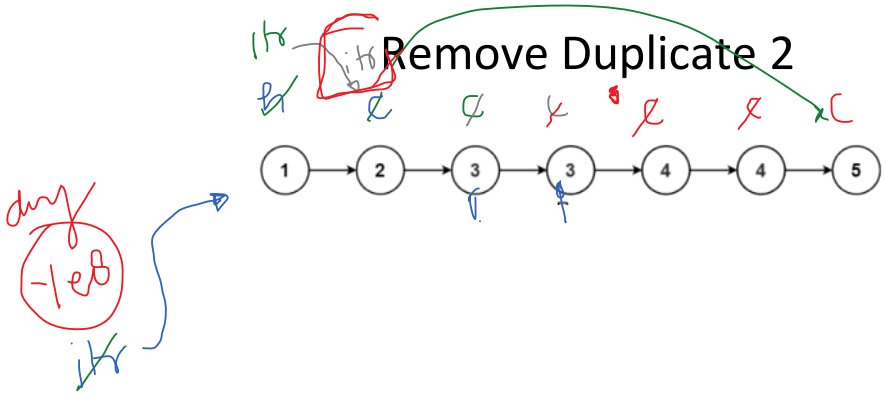
Remove Duplicates from Sorted List



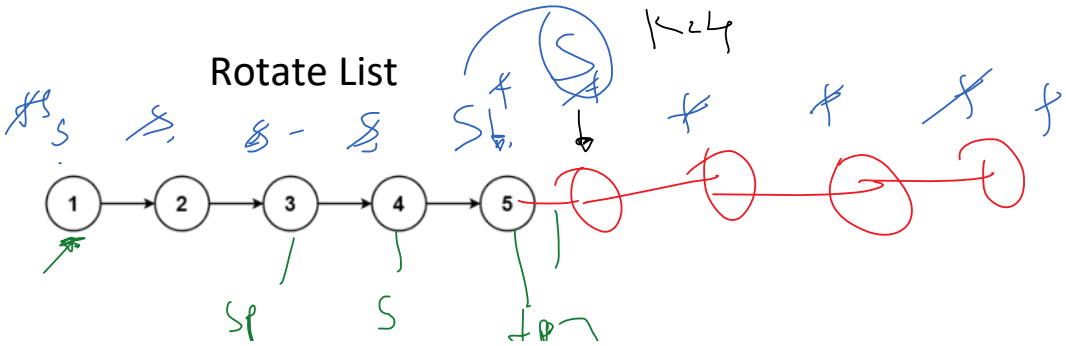


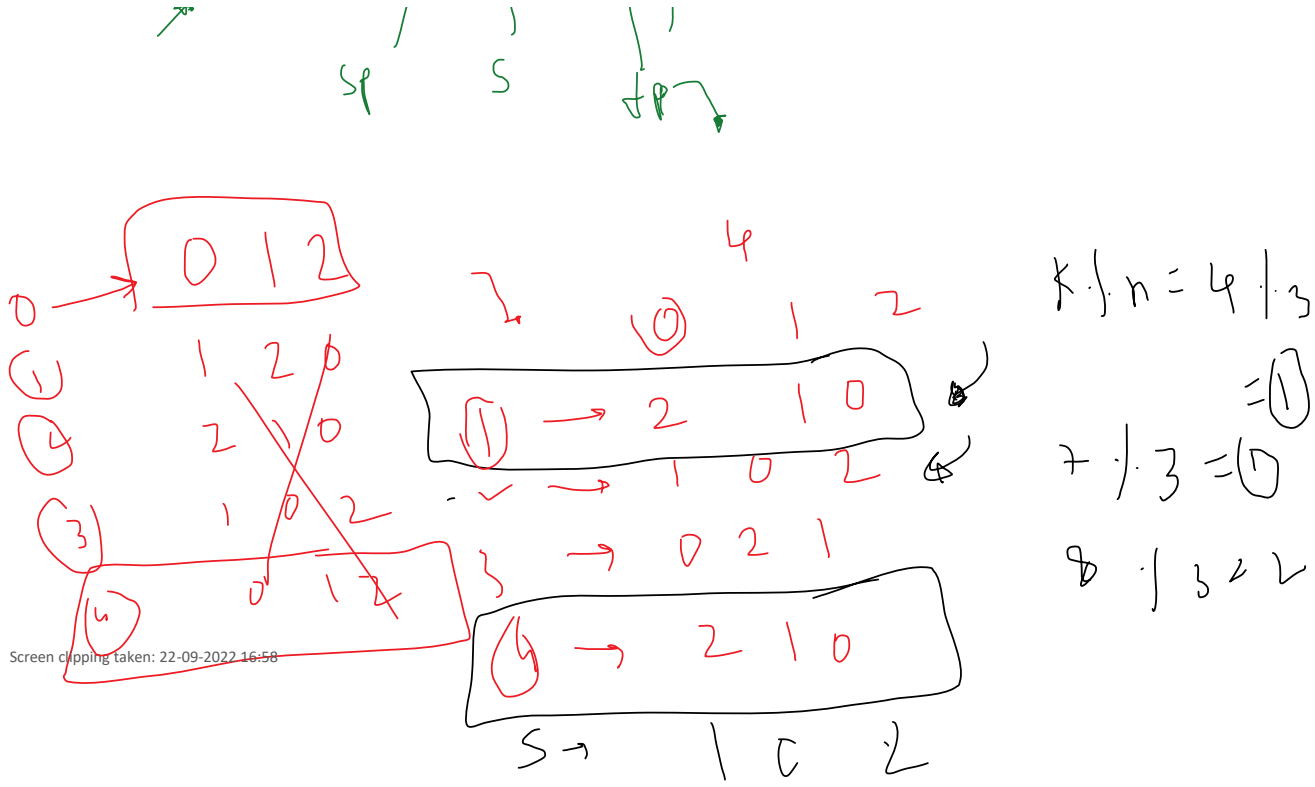

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head)
    {
        ListNode * dummy = new ListNode(-1e8);
        ListNode * itr = dummy;
        ListNode * curr = head;
        while(curr != NULL)
        {
            if( itr->val == curr->val )
            {
                curr = curr -> next;
            }

            itr-> next = curr;
            itr = itr -> next;
            if(curr != NULL) curr = curr->next;

        }
        return dummy-> next;

    }
};
```

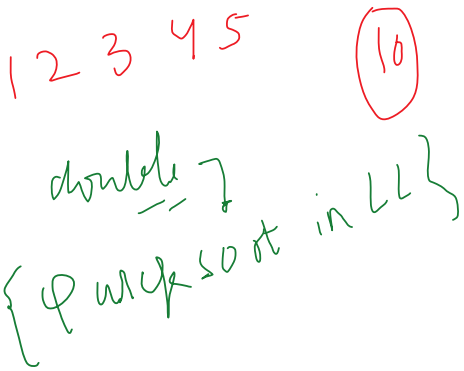
Remove Duplicate 2



Rotate List





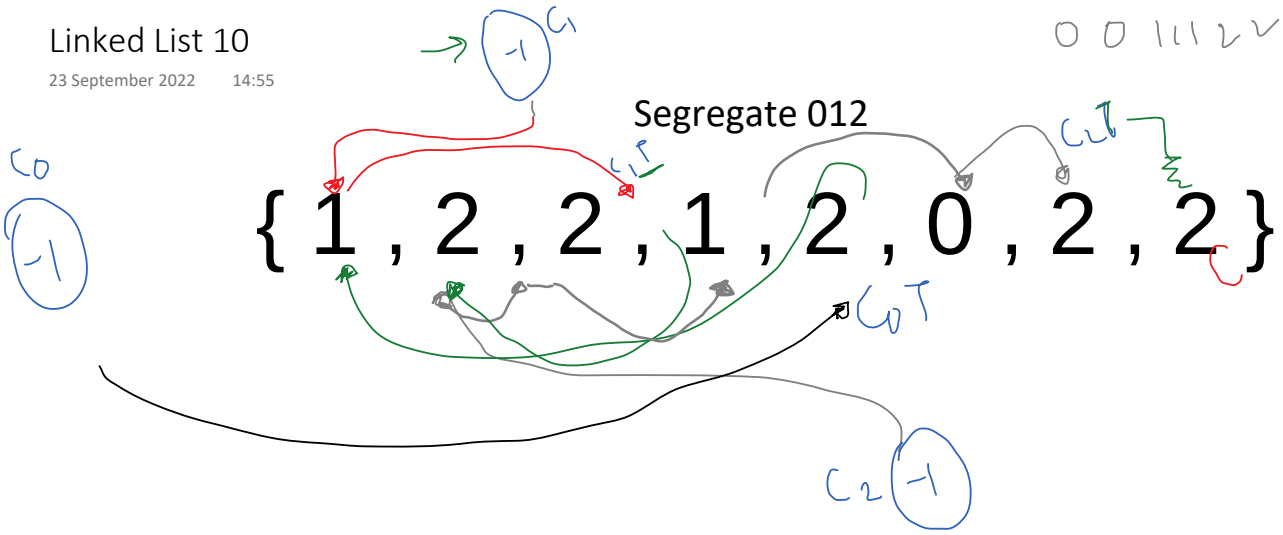


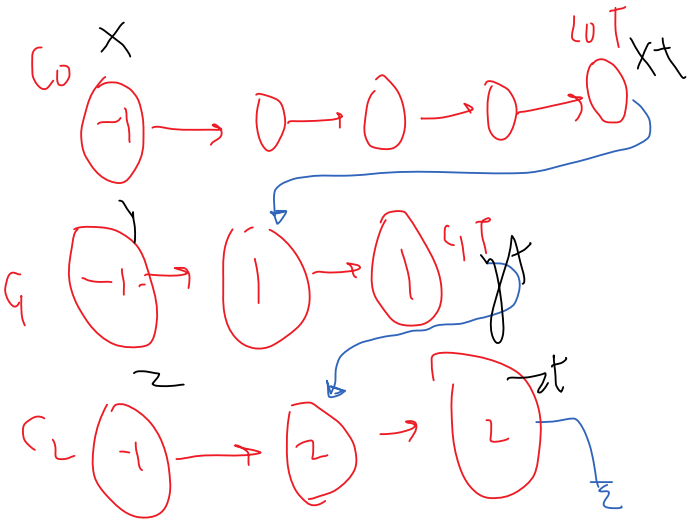
```
ListNode* rotateRight(ListNode* head, int k)
{
    int n = getsize(head);
    if(head == NULL || head-> next == NULL || k == 0 || k == n ||  ) return head;


    k = k % n  ;
    ListNode * slowprev = NULL;
    ListNode * slow = head;
    ListNode * fast = head;
    ListNode * fastprev = NULL;

    while(k -- && fast != NULL)
    {
        fast = fast -> next;
    }
    while(fast != NULL)
    {
        slowprev = slow;
        slow = slow -> next;
        fastprev = fast;
        fast = fast -> next;
    }
    if(slowprev != NULL)
        slowprev -> next = NULL;
    fastprev -> next = head;
    return slow;


}
```

Linked List 10

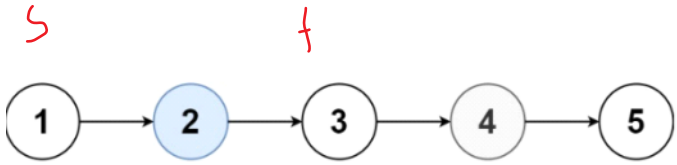
23 September 2022 14:55

Segregate 012





Kth Node form the end



Reverse in K groups