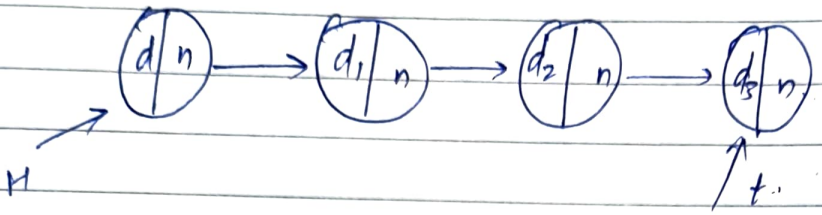
APCO
Date :
Page :

1. Middle of linked list ① ceil ② Floor
2. Reverse a linked list (pointer).
3. Reverse a linked list (data).
4. Palindromic linked list // 234
5. 143 // Reorder list
6. Reorder → normal.
7. Merge 2 sorted list
8. Merge k sorted list
9. Reverse nodes in k groups (25)
10. 148 // sort list
11. 92 // Reverse b/w.
12. 141 // cycle detection.
13. 142 // ll cycle II
14. 160 Find intersectionpoint
15. 61
16. 19
17. Segregate even and odd.

Date.....

Linked List. 09/09/2020

→ avoid recursion in ll. question.



Should

1. throw exception.
2. Code readable.
3. Transfer कर सकेगा
4. code adaptive होना चाहिए

Function of stl.

	cpp.	Java.
1	Push_front().	addFirst()
2	push_back().	addLast().
3	push_at().	addAt().
4	pop_	pop removeFirst().
5	pop_	removeLast()
6	pop_	removeAt()
7	getx().	getx()
8	size().	size()
9	isEmpty()	isEmpty

Should have to write industry level code.

Spiral

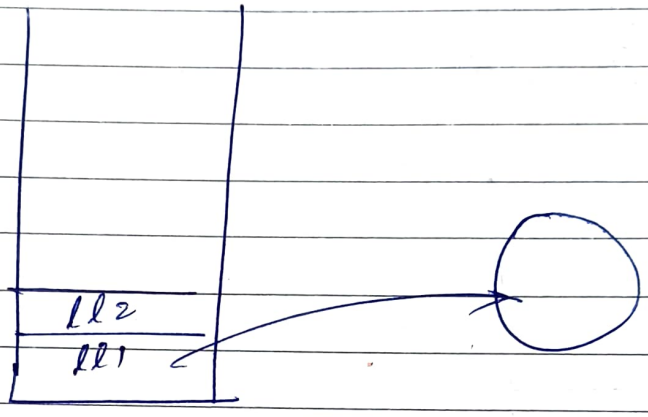
Date.....

Array. 

Contiguous memory location.

LL. 

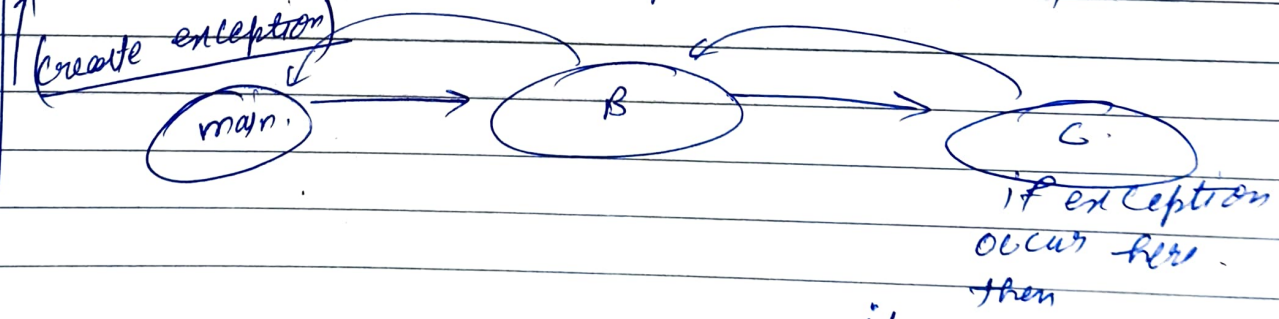
Random memory location.



this object को refer करता

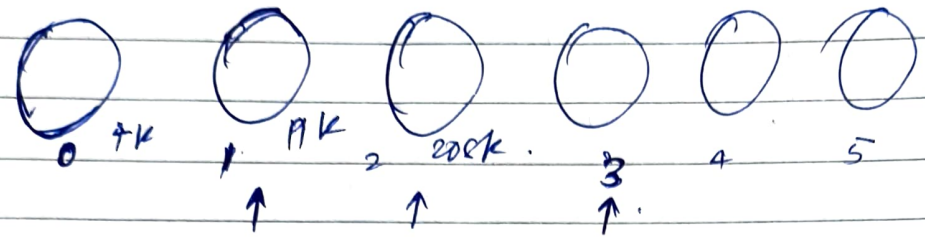
★ throws :- fn के साथ capability है exception throw करने की

throw :→ particular exception को defined

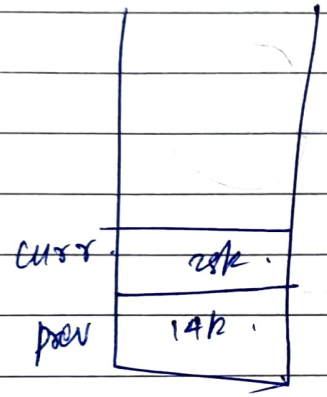


Date.....

Q. ~~Linked~~ It very easy topic and qn आते हैं interview में.



idx - -> 0
curr head पर है



Node (prev) = getNode(idx-1).

→ pointer and only exist at stack not any relation with heap.

Spiral

Date.....

class बनाते है qn का तरीका interview oops के concept मांगता चाहता है interviewer.

① Stack, queue, Unfixed list
↓ ↓ ↓
private, public, enclapsulation
means oops के concept का सारा use करते

very easy है U के question आते है.

① a → 3.
② r → 3.
③ g → 3.
④ size, empty

① If modification allowed. →

(421)

Date.....

(11-09-220)

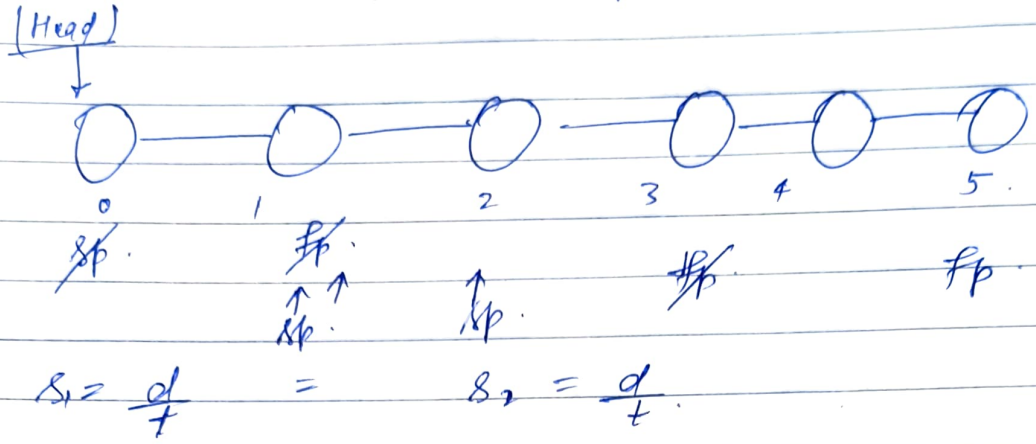
(Linked list question)

Agenda-

1. Middle of linked list → a) ceil b) floor. // 876.
2. 206 l.c // Reverse a linked list pointer. (3 pointer app).
3. Reverse a linked list (data only).
4. Pallindromic linked list /// 234 leetcode
5. 143 (lc) Reorder list.
6. Reorder list → normal.
7. leetcode - 21. (Merge two sorted list)
8. Leet Code - 23. /// Merge k sorted list.

Date.....

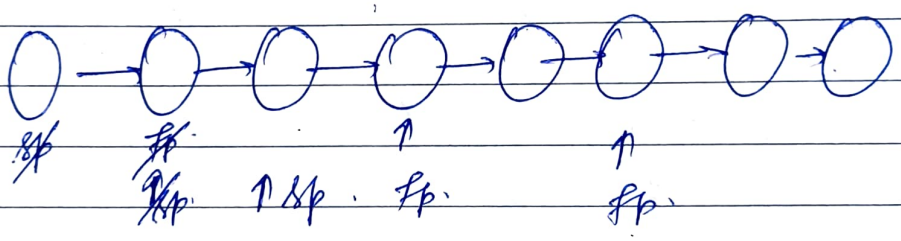
Q) Middle of Linked list (M-1 two pointer). [Leet 876]



$s_1 = \frac{d}{t}$ = $s_2 = \frac{d}{t}$.

ds:

2) even - Case.



* Choose first mid in even case.

* if qn asked about mid then tell 2nd mid.

M-1 Ceil node of middle.

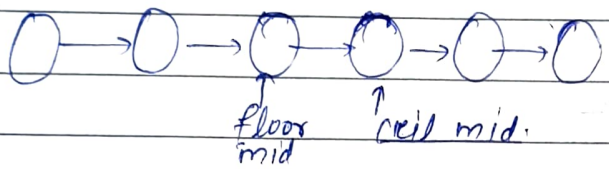
```
ListNode* middleNode(ListNode *head){
    if(head==nullptr || head->next==nullptr){
        return head;
    }
    ListNode *slow = head;
    ListNode *fast = head;
```

Date.....

```
while( fast != nullptr && fast->next != nullptr){
    slow = slow->next;
    fast = fast->next->next;
}
return slow;
```

T: O(n).
S: O(1).

M-2 it gives floor mid eg.



if anyone ask mid → ceil is mid.
but for question floor mid is imp. and if we need ceil we can give next of our ans.

```
ListNode* middleNode2(ListNode *head){
    if(head == nullptr || head->next == nullptr){
        return head;
    }
    ListNode* slow = head;
    ListNode* fast = head;

    while(fast->next != nullptr && fast->next->next != nullptr){
        slow = slow->next;
        fast = fast->next->next;
    }
    return slow;
}
```

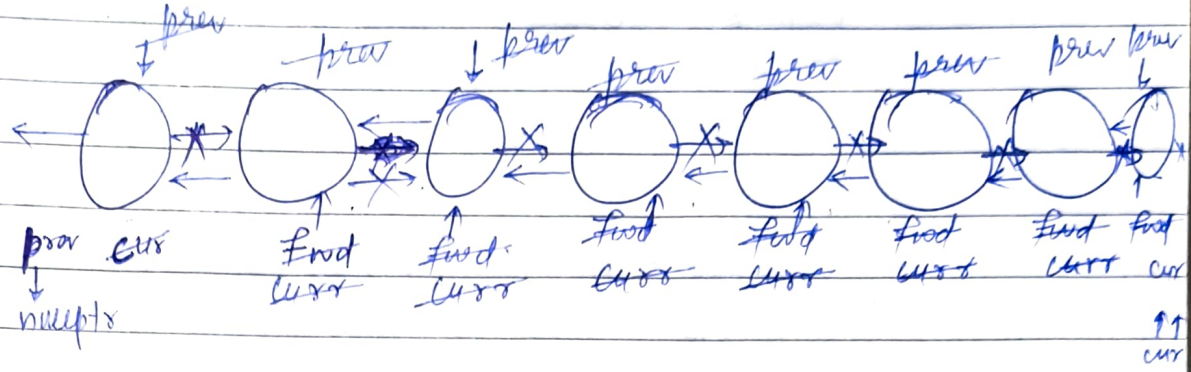
T: O(n)
S: O(1)

Spiral

cpp		java
→	=	.
Node*	=	Node
nullptr	=	null

Date.....

Qu. // leetcode (206). Reverse a linked list



Jump

```
if(head == nullptr && head->next = nullptr) {
    return head;
}
```

add this line to every linked list code.

// Code -

```
ListNode* reverseList(ListNode* head) {
    if(head == nullptr || head->next == nullptr){
        return head;

    ListNode* prev = nullptr;
    ListNode* fwd  = nullptr;
    ListNode* curr = head;

    while( curr != nullptr) {
        fwd  = curr->next ; // backup
        curr->next = prev ; // Link.
        prev = curr;        // move.
        curr = fwd ;
    }
    return prev;
}
```

Spiral

Date.....

// Code of ReverseList data Only.

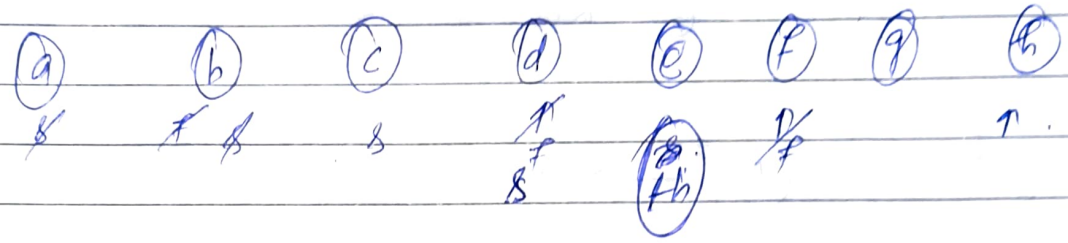
```
void ReverseData (ListNode *head) {
    if (head == nullptr || head->next == nullptr)
        return;

    ListNode* mid = middleNode2(head); // 2nd middle
    ListNode* thead = mid->next;
    mid->next = nullptr;

    thead = reverseList(thead);
    ListNode* c1 = head;
    ListNode *c2 = thead;

    while(c1 != nullptr && c2 != nullptr){
        int temp = c1->val;
        c1->val = c2->val;
        c2->val = temp;

        c1 = c1->next;
        c2 = c2->next;
    }
    thead = reverseList(thead);
    mid->next = thead;
}
```

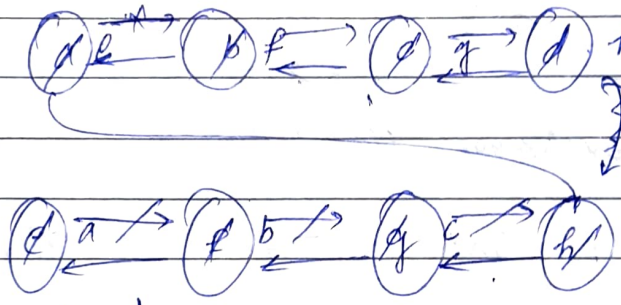
Spiral

Date.....

* don't use linear recursion in LL. becoz it's best to array h.

Q: Reverse LL in data only.





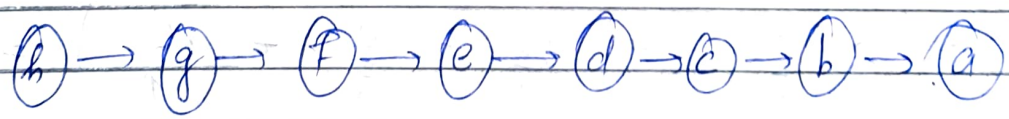
(temp head)

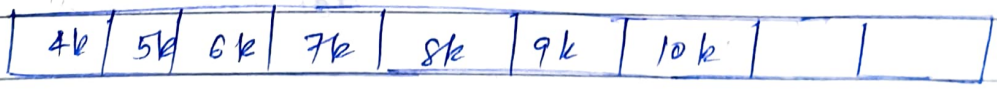
mid से आगे से. n/2 other mid से Reverse करते जो n/2 data change हो to make same as before.

$\left(\frac{n}{2} + \frac{n}{2} + \frac{n}{2} + \frac{n}{2}\right) = 2n \Rightarrow O(n)$.

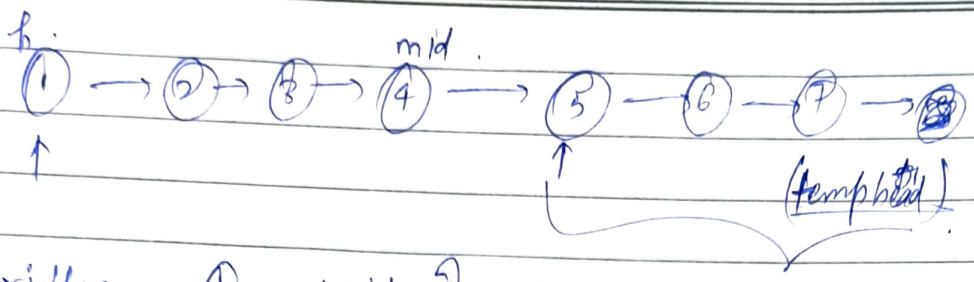
Time = $O(n)$.

*





Date.....

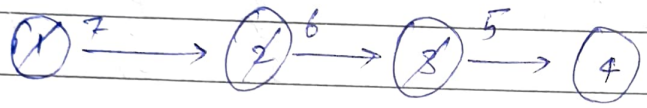


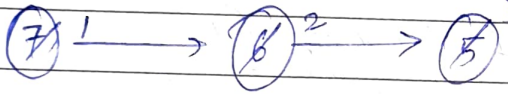
Algorithm: ① mid निकालो mid.

② mid का next निकालो ⇒ thead.

③ Reverse किया thead वाले portion को

eg =



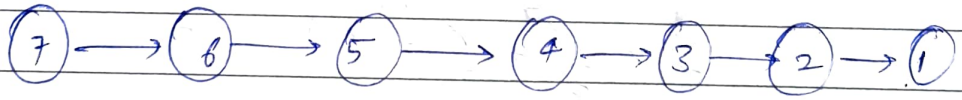


④ आपस में exchange the values

⑤ फिर से reverse list call करके only for thead portion.

⑥ mid → next = thead वाला किया

eg =



(Reverse only data).

T: $O\left(\frac{n}{2} + n/2 + \frac{n}{2} + n/2\right) = O(2n)$

↑ mid निकाला ↑ reverse ↑ वापस change again reverse ↑

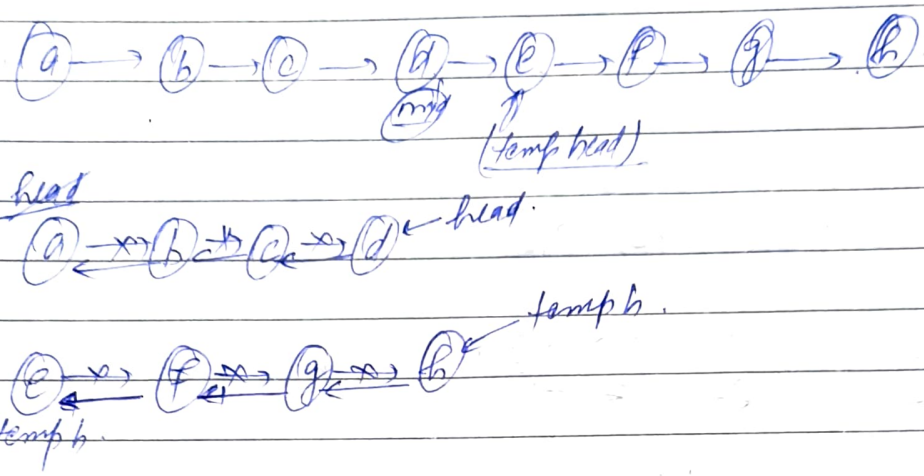
$T = O(n)$

$S : O(1)$.

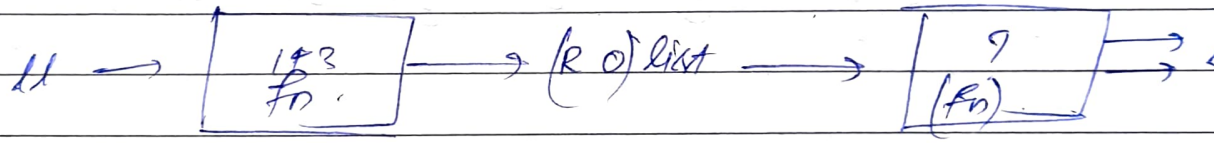
Spiral

Date.....

Qn. Palindrome Linked List // 234 (LeetCode)

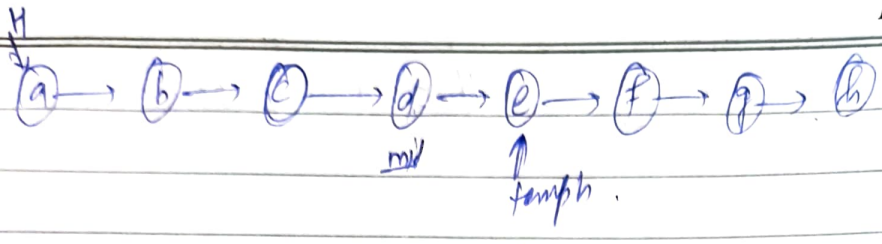


143 / Reorder list,



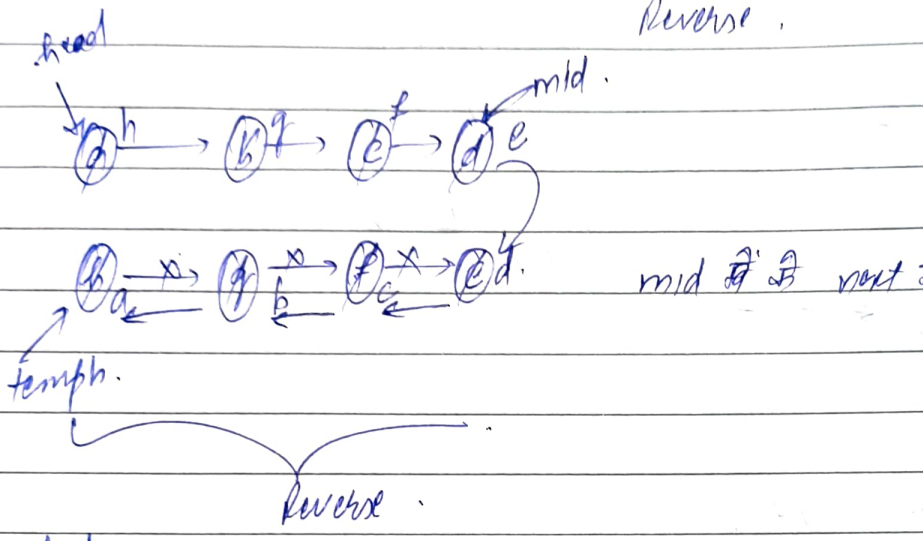
O(n) Time
S : O(1).

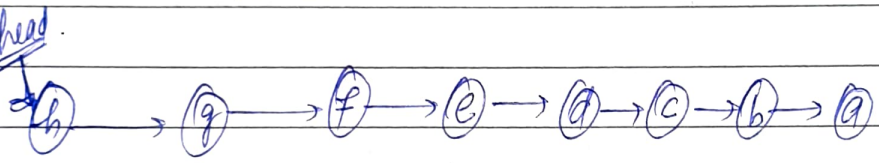
Date.....



prev qn.

dry run of reverse data linked list)





Note Same approach as of prev qn instead of changing data in two half linked list just compare data and if not match return false.

Date.....

```
bool isPalindrome(ListNode* head) {
    if(head == nullptr || head->next == nullptr){
        return head, true;
    }

    ListNode* mid = middleNode2(head);
    ListNode* tHead = mid->next;
    mid->next = nullptr;

    tHead = reverseList(tHead);
    ListNode* c1 = head;
    ListNode* c2 = tHead;

    bool res = true;
    while(c1 != nullptr && c2 != nullptr){
        if(c1->val != c2->val){
            res = false;
            break;
        }

        c1 = c1->next;
        c2 = c2->next;
    }
    tHead = reverseList(tHead);
    mid->next = tHead;
    return res;
}
```

T: $O(n/2 + n/2 + n/2) = 3 \times \frac{n}{2} = O(n)$.
(↑ mid, ↑ reverse, ↑ compare)

S: $O(1)$.

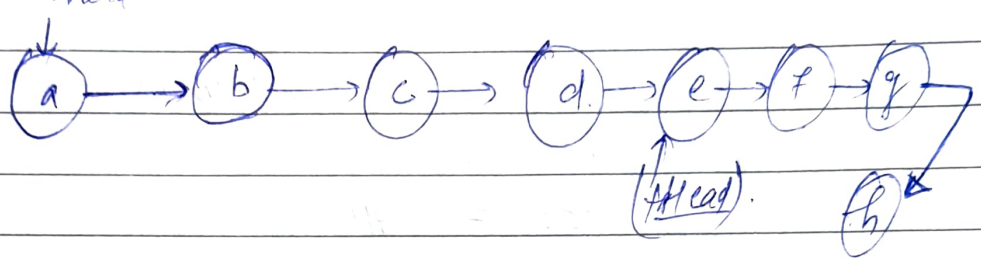
Spiral

Date.....

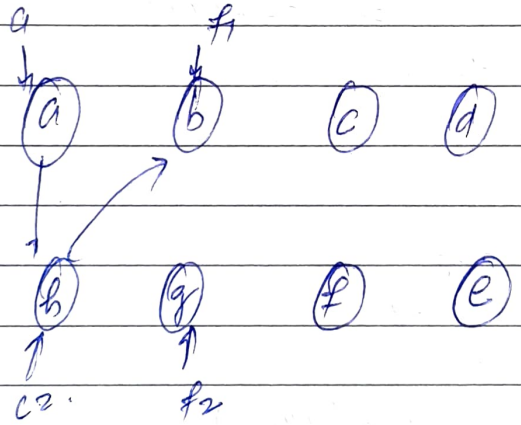
Note: Always dry run for ① even ② odd ③ single & 3 elements के लिए

Qn 143 (Reorder List).

Dry Run:-



① middle निकालना
② middle का next को ttHead बनाना
mid→next = nullptr;



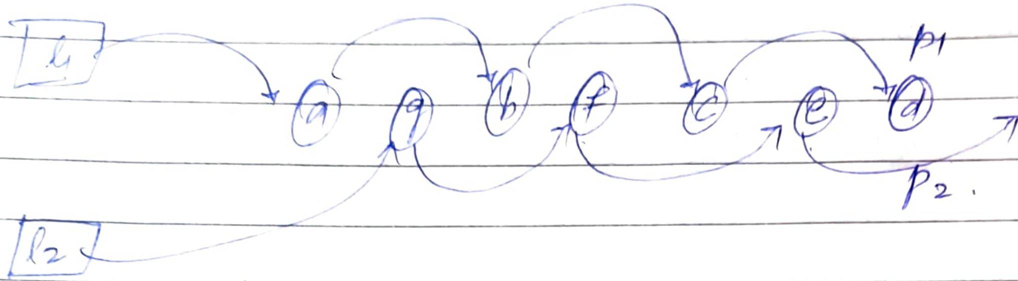
Code /// = void ReorderList (ListNode *head) {

```
Void reorderList (ListNode* head) {
    if (head == nullptr || head->next == nullptr)
        return;
    ListNode* mid = middleNode2 (head);
    ListNode* ttHead = mid->next;
    mid->next = nullptr;
}
```

Spiral

Date........................

Qu Reorder list → Normal Linked list



(reverse (p2) and add it to p1). T: O(n).
S: O(1).

Best and easy
M-2 temp H1 → (a) → (b) → (c) → (d)

temp t1 → (e) → (f) → (g)

T: O(n).
S: O(1).

Note: ये Question fixed है । इसमें and very easy to understand है so अच्छे से करो

prev 143

```
tHead = reverselist (tHead);
ListNode* c1 = head;
ListNode *c2 = tHead;

ListNode* f1 = nullptr;
ListNode *f2 = nullptr;
```

Date.....

Code को करते जिसमें mistake के chance कम हो क्यों कि सामने वाला का code अगर ठीक है तो आप निकाल दिए जाएंगे).

(आपका competition lies in interviews)

c1, c2 → current
↓
current1

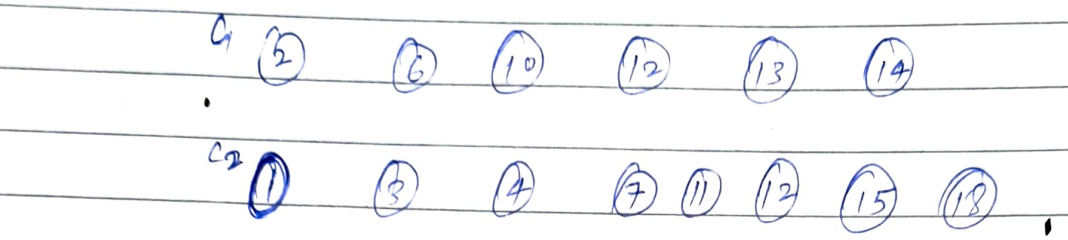
// 143

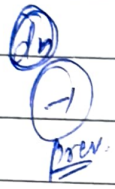
```
while(c1 != nullptr && c2 != nullptr){
    f1 = c1->next; // backup.
    f2 = c2->next

    c1->next = c2; // link.
    c2->next = f1;

    c1 = f1;      // move.
    c2 = f2;
}
```

} fn का loop.

Date.....

// Leetcode - 21 Inplace (without extra space).
(Merge two sorted List)





dp, bit

① Array, String.
② Range tree,
③ dp

T: O(n).
S: O(1).

643 Div 2, B

Code //.

```
ListNode* mergeTwoLists(ListNode *l1, ListNode* l2){
    if(l1==nullptr || l2==nullptr){
        return l1==nullptr ? l2 : l1;
    }

    ListNode* dummy = new ListNode(-1);
    ListNode * prev = dummy;

    ListNode *c1 = l1;
    ListNode *c2 = l2;
```

Date.....

```
while ( c1 != nullptr && c2 != nullptr )
    if( c1->val < c2->val ) {
        prev->next = c1;
        c1 = c1->next;
    }
    else {
        prev->next = c2;
        c2 = c2->next;
    }
    prev = prev->next;
    if ( c1 != nullptr ) {
        prev->next = c1
    else
        prev->next = c2;

    return dummy->next;
}

} // Fn end.
```

Date.....

(13/09/2020)

(Linked List Question)

Agenda

1.

2. Merge k sorted list (23).

Reorder → normal list (2nd part of 143).

```
ListNode *th1 = nullptr;
ListNode *t1 = nullptr;
ListNode *th2 = nullptr;
ListNode *t2 = nullptr;

fn 1
Void addFirst(ListNode *node){
    if(th1 == nullptr){
        th1 = node;
        t1 = node;
    }
    else{
        node->next = th1;
        th1 = node;
    }
}

fn 2
Void addLast(ListNode* node){
    if(th2 == nullptr){
        th2 = node;
        t2 = node;
    }
}
```

Spiral

Date.....

```
    else {
        th2->next = node;
        th2 = node;
    }
}  // fn;
```

Main fn is Now writing which will use addFirst and addLast().

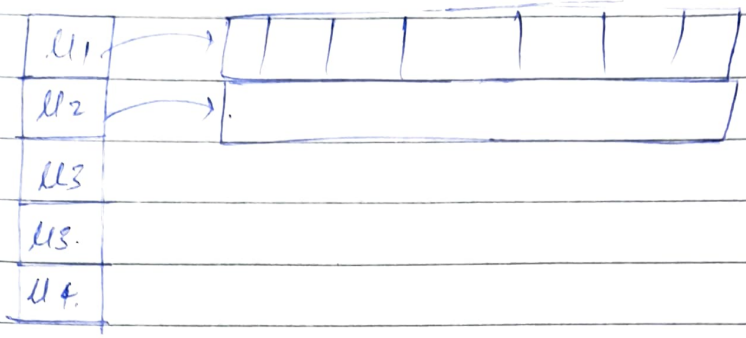
// Code -

```
ListNode* orderList (ListNode* head) {
    if (head == nullptr || head->next == nullptr)
        return;

    int cnt = 0;
    ListNode* curr = head;
    while (curr != nullptr) {
        ListNode *fwd = curr->next;
        curr->next = nullptr;

        if (cnt = 0) {
            addLast (curr);
        }
        else {
            addFirst (curr);
        }
        curr = fwd;
        cnt = (cnt+1) % 2;
    }
    th2->next = th1;
    return th2;
} // fn
```

Spiral

Date.....

Ans 23:



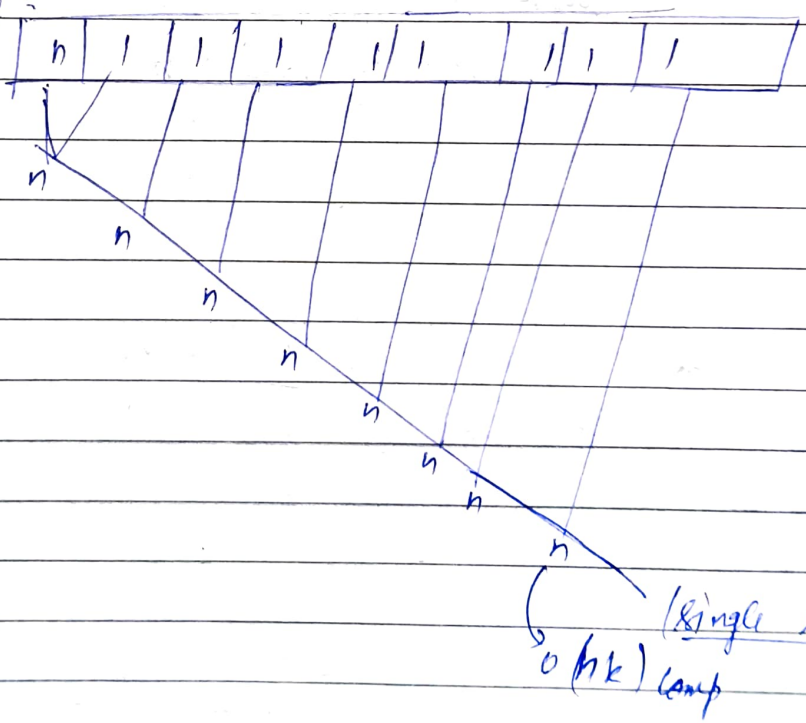
M-1 Heap (Priority of Comparator & pointer).

T: O(n log k).

$\sum n1 + n2 + n3 + \cdots + nn = n$.

k = no of linked list / size of array.

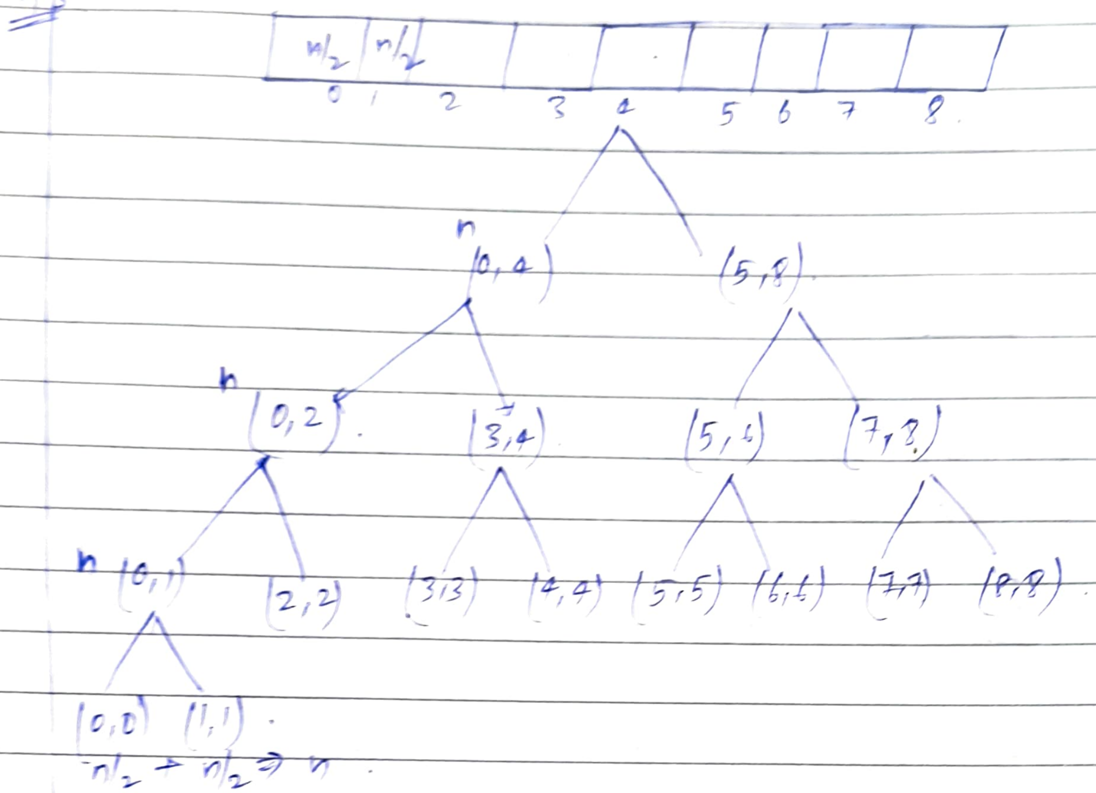
M-2 Brute force.



(kn)

Date........................

M-S -



(logk) n.
→ (height of tree) × n

① mid 2
② Reverse
③ Merge 2
④ addFirst / addLast

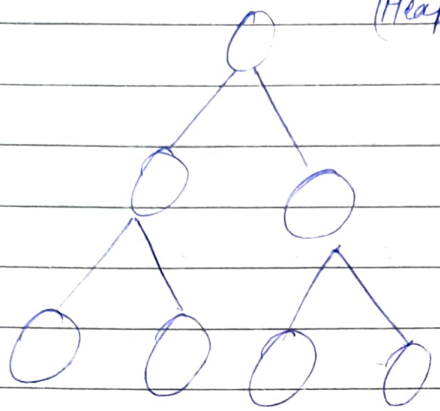
(enough for solving any linked list code)
Rate होने चाहिए all these fn

n logk <<< n logk.
fast fast होता है

Date.....

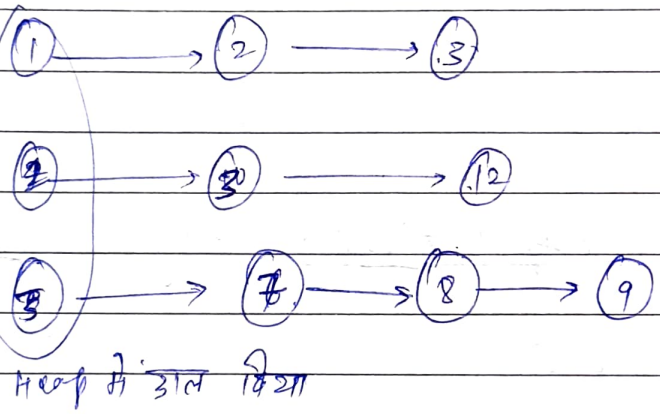
Note :-

log(n) fast → searching fast होता है
∵ of continuous memory location
(Heap) में logn से search.



(logn)

M1.



Heap में डाल दिया

prev → (1)

1, 2, 3

when arr → 104 / 103 then.
array तभी perform better than heap.
M3 better than M1

Date.....

Soln: //Code
M-3 - //23

```
ListNode* mergeKLists(vector<ListNode*>& lists, int si, int ei){
    if(si==ei){
        return lists[si];
    }
    int mid = (si+ei)/2;
    return mergeTwoLists(mergeKLists(lists, si, ei),
                         mergeKLists(lists, mid+1, ei));
}
```

// Merge K List heap Methode: using ~~comparartor~~

```
class Compare {
public:
    bool operator()(const ListNode *a, const ListNode *b
                    const {
        return a->val > b->val; // this - other
                                default;
    }
};
```

Date..........................

```
ListNode * mergeKLists(vector<ListNode*>& lists){

    if(lists.size()==0){
        return nullptr;
    }
```

↓ type ↓ Contain of that type

```
    priority_queue<ListNode *, vector<ListNode*>,
                   compare> que;
```

↑ बताता है min बनाना है max बनाना है etc. heap

```
    for(int i=0; i<lists.size(); i++){
        que.push(lists[i]);
    }

    ListNode* dummy = new ListNode(-1);
    ListNode* prev = dummy;

    while(que.size() != 1){
        ListNode *node = que.top();
        que.pop();

        ListNode *next = node->next; //backu
        node->next = nullptr;

        prev->next = node; //link;
        prev = node;

        if(next != nullptr){
            que.push(next); // move
    }
```

Spiral

Date.....

```
        prev->next = que.top();
        que.pop();

        return dummy->next;
    }
```

Spiral

Priorityqueue ∠ Type,

Date
Page

(13/09/2020)

1. mid 2 → चलना ही चाहिए
2. reverse
3. Merge 2 ll.
4. addfirst/addLast.

नोट में भी आना चाहिए
& first attempt में
चलना चाहिए

25. ~~Ex~~ Agenda:

1. 25 // Reverse nodes in k groups.
2. 148 // sort List.
3. 92 //

Derivation for Complexity

Merge sort of two linked list Time complexity derivation.

$$T(n) = T(n/2) + T(n/2) + n/2 + n$$

($T(n/2)$: half list. $T(n/2)$: another half. $n/2$: mid. n: merge करने के लिए.)

$$T(n) = 2T(n/2) + 3n/2$$

$$2T(n/2) = 2^2T(n/4) + 2\left(\frac{3n}{2}\right) \times \frac{1}{2}.$$

$$2^2T(n/4) = 2^3T(n/8) + 2^2\left(\frac{3n}{2}\right) \times \left(\frac{1}{4}\right).$$

$$2^3 \times T(n/8) = 2^4T(n/16) + 2^3\left(\frac{3n}{2}\right) \times \left(\frac{1}{8}\right).$$

$$T(n) = 2^k T(0) + \left(\frac{3n}{2}\right) k,$$

$k = \log n$ (because doing half every time).

$$T(n) = 2^{\log_2 n} + \left(\frac{3n}{2}\right) \log_2 n.$$

$$\boxed{T(n) = n + \frac{3n}{2} \log_2 n.}$$

APCO
Date :
Page :

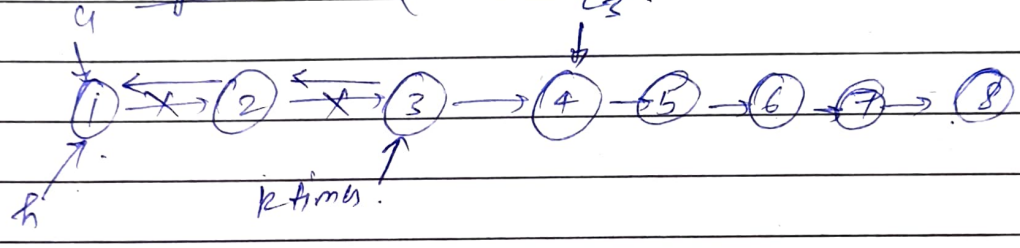
** Priority queue (Type , Container < type > , greater < type >)

only for primitive data type like, int, char, vector etc. work

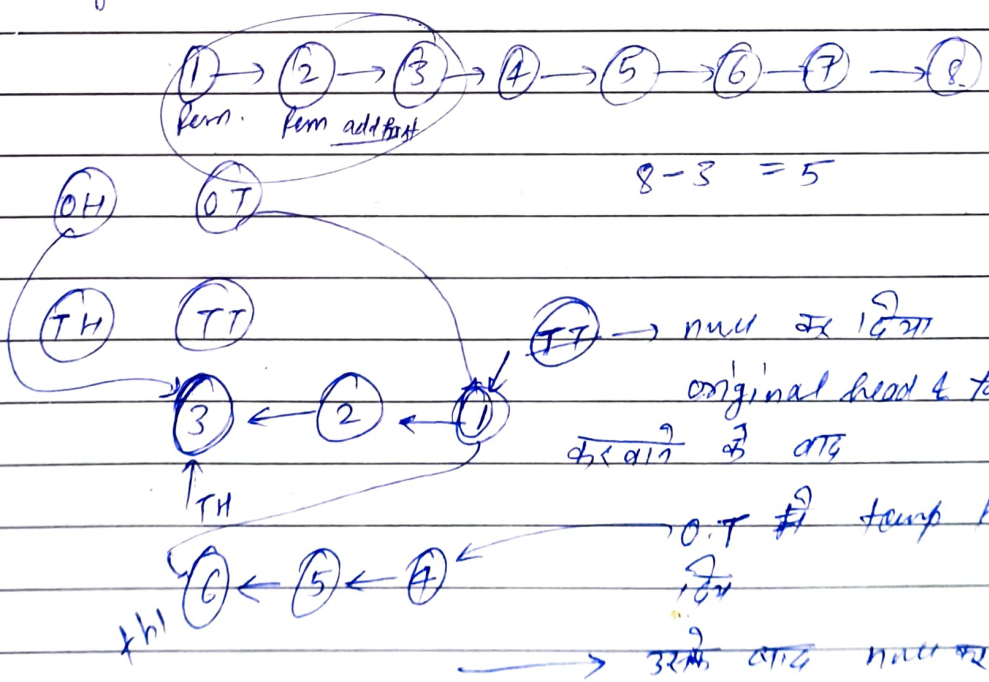
But for Char, have to write compare class and perform whatever whether min/max heap or that is needed.

25 Dec Dummy Node. (Reverse k group).

M-1



M-2 Using addFirst and addLast



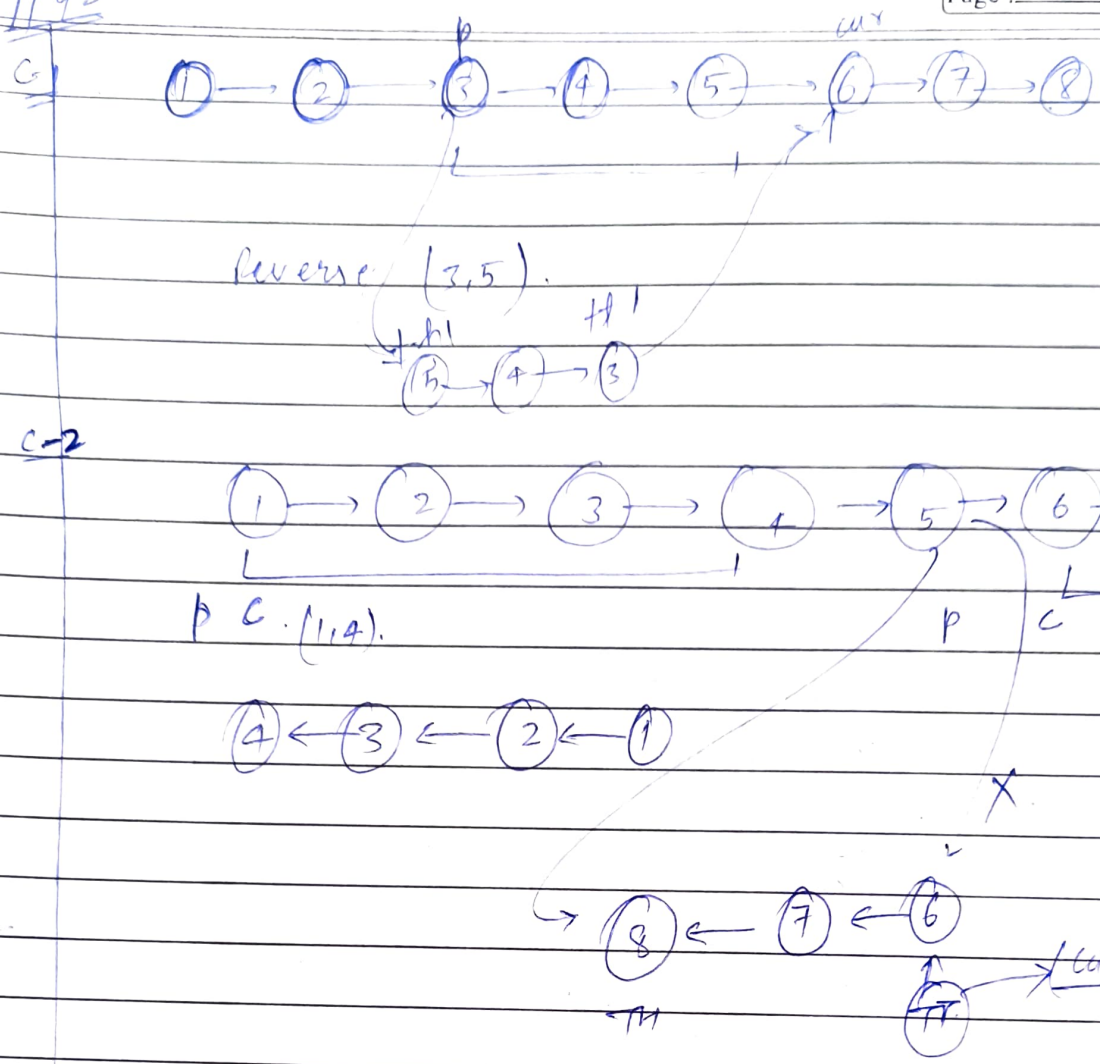
APCO
Date :
Page :

M-2 // Code :

```
int len (ListNode * head){
    int l = 0;
    while (head != nullptr){
        l++;
        head = head -> next;
    }
    return l;
}
ListNode * reverseKGroup (ListNode *head, int k){
    if(head == nullptr || head -> next == nullptr)
        return head;
    int l = len(head);
    if(l < k)    return head;

    ListNode *curr = head;

    ListNode *nh = nullptr;
    ListNode *ot = nullptr;

    int k = k;
    while (curr != nullptr && l >= k){
        while (k-- > 0){
            ListNode* next = curr -> next;
            curr -> next = nullptr;

            addFirst(curr);
            curr = next;
        }
```

APCO
Date :
Page :

// 92



APCO
Date :
Page :

// 25 Continue-

```
        if (oh == nullptr) {
            oh = th1;
            ot = th1;
        }
        else {
            ot -> next = th1;
            ot = th1;
        }

        th1 = nullptr;
        th1 = nullptr;

        l1 = k;
        k = k;
    }
    outer while loop)
    ot -> next = curr;
    return oh;
}
```

APCO
Date :
Page :

// leetcode 92 :-

```
ListNode * reverseBetween(ListNode *head, int m, int n){
    if(head == nullptr || head->next = nullptr || m==n)
        return head;

    ListNode *curr = head;
    ListNode *prev = nullptr;

    int idx = 1;
    while(curr != nullptr){

        while(idx >= m && idx <= n){

            ListNode *next = curr->next;
            curr-> next = nullptr;

            addFirst(curr);
            curr = next;
            idx++;
        }

    if(idx > n){
        if(prev == nullptr){
            head = th1; }
        else
            prev->next = th1;

        th1->next = curr;
        break;
    }
```

used to reverse the given list.

assign head & tail for diff cases.

```
        prev = curr;
        curr = curr->next;
        idx++;
    }   // outer while loop.
    return head;
}
```

move till condition becomes true.

Qu leetcode 148: Sort List

4 → 2 → 1 → 3.
1 → 2 → 3 → 4 (sort).

I/P = -1 → 5 → 3 → 4 → 0
O/P = -1 → 0 → 3 → 4 → 5

```
ListNode* sortLists(ListNode *head){
    if(head==nullptr || head->next==nullptr){
        return head;
    }
    ListNode *mid = middleNode2(head);
    ListNode *nhead = mid->next;
    mid->next = nullptr;

    return mergeTwoLists(sortList(head), sortList(nhead));
}
```

Date :
Page :

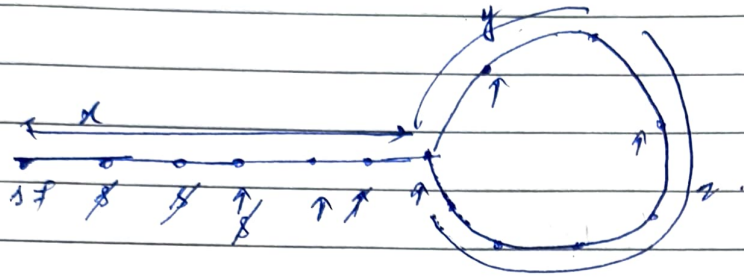
(14/09/2020) (Linked List)

Agenda:

1. cycle detection. || 141 leetcode.
2. 142 || linked list cycle II
3. 160 find intersection point
4. 61
5. 19.
6. Segregate even and odd.

APCO
Date :
Page :

①



$s_1 = \frac{d_1}{t} \Rightarrow t = \frac{d_1}{s_1} \Rightarrow \frac{d}{s}$.

$s_2 = \frac{d_2}{t} \Rightarrow t = \frac{d_1}{s_2} = \frac{d}{2s}$.

$t = t = \frac{d}{v} \Rightarrow$

$d = vxt$.

$d = (2vt)$.

derivation :-

(d_s = dist travel by slow pointer).

d_f = dist travel by fast pointer).

$s_s = \frac{d_s}{t_s}$, $s_f = \frac{d_f}{t_f}$

$d_s = (x+y)$, $d_f = x+(y+z)m+y$.

$t_s = t_f$.

$$\frac{x+y}{s_s} = \frac{x+(y+z)m+y}{s_f}.$$

$$x(x+y) = x+(y+z)m+y.$$

$$(x+y)(x-1) = (y+z)m.$$

$$\boxed{(x+y) = (y+z)\frac{m}{(x-1)}}^{*}.$$

$x+y \neq 0$
$y+z \neq 0$
$m > 0$

$\boxed{x > 1}^{*}$

$\boxed{\frac{sf}{ss} > 1}$ ★ $\boxed{r \in N}$ ★ $\forall r$

2, 3, 4, 5 ...

C-2 slow ने कुछ circular कर लिया

$$ds = (x+y) + (y+z)n$$

$$r(x+y) + r(y+z)n = (x+y) + (y+z)m$$

$$(r-1)(x+y) = (m-nr)(y+z) \quad \text{— ①}$$

$$(x+y) = \frac{(m-nr)}{r-1}(y+z)$$

becoz if same speed है तो initially p से next point पर ही fast आगे होगा	$r-1=0$ $r=1$ $(r \neq 1)$ ∵	$m-nr=0$ $m=nr$ $r=\frac{m}{n}$	$\frac{x+y}{y+z} = -n$
	$r-1>0$ $r>1$	$m-nr>0$ $\boxed{\frac{m}{n} > r > 1}$ ★ $\frac{m}{n} > r > 1$	

Cycle finding
(Floyed ~~warshat~~ algorithm)

Case 2 $r-1>0$
$r>1$

$m-nr>0$

$\boxed{\frac{m}{n} > 1}$ ∵ n se independent

$m > 1 \times n$

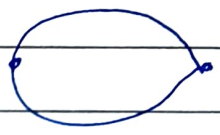
$\boxed{m>0}$ min to meet slow and fast

fast have to make atleast one rotation.

APCO
Date :
Page :

// code - leetcode (141).

```
bool hasCycle (ListNode *head) {
    if(head == nullptr || head->next == nullptr){
        return false;
    }

    ListNode* slow = head;
    ListNode* fast = head;

    while( fast != nullptr && fast->next != nullptr){

        slow = slow->next;
        fast = fast->next->next;
        if (slow == fast)
            break;
    }
    return slow == fast;
}
```



APCO
Date :
Page :

Qu 142 // Linked list cycle 2.

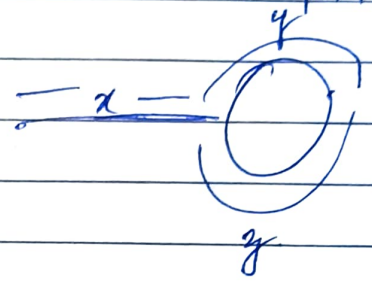
we have

$$(r-1)(x+y) = (y+z)m.$$

if $(r=2)$.

$$x+y = (y+z)m.$$

(here 'm' is rotation of fast pointer)

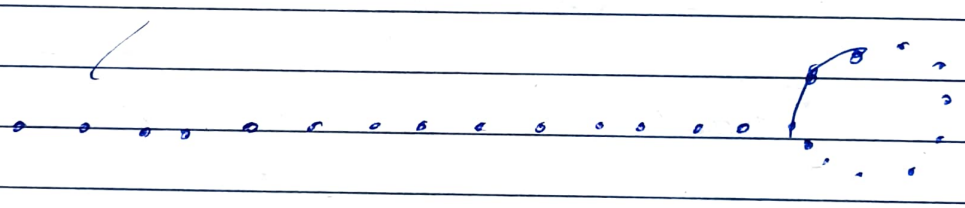


$m=1$

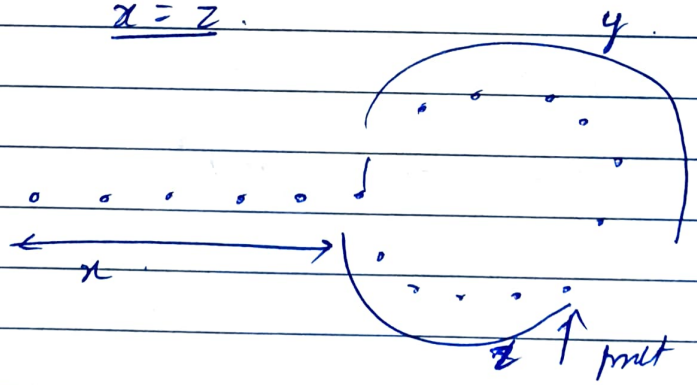
$x = z.$

$m=2$

$$\boxed{x = z + (y+z);}$$ ✱



$m=1$ $x=z.$

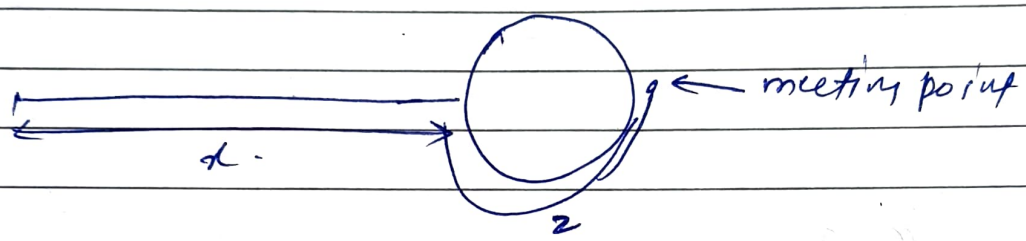


APCO
Date :
Page :

$$x = y + (y+z)\,m - 1$$

m denotes cycle.

m=1 मतलब 1 rotation के



x

$$x = z + (y+z)\,m - 1$$

slow x dist travel करता है

fast (m-1) rotation + z count travel करता है. meet.

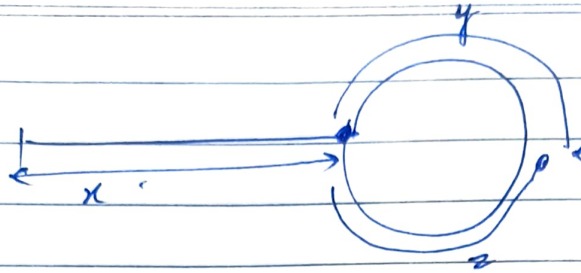
~~$(x-1)(x+y) = (y+z)\,m$~~

$$x = y + (y+z)(m-1)$$

$m = 1$

$x = y$. (means 1 rotation में y point move करते हैं meters में)

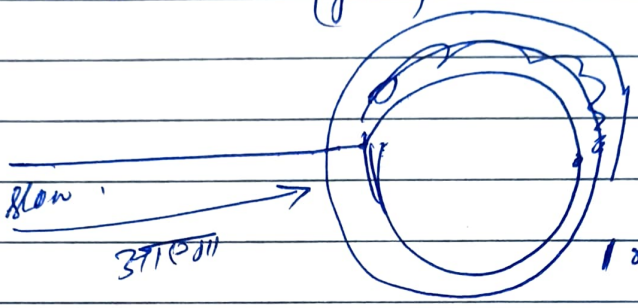
APCO
Date :
Page :



2 point move

m=2.

$x = z + (y+z)$.



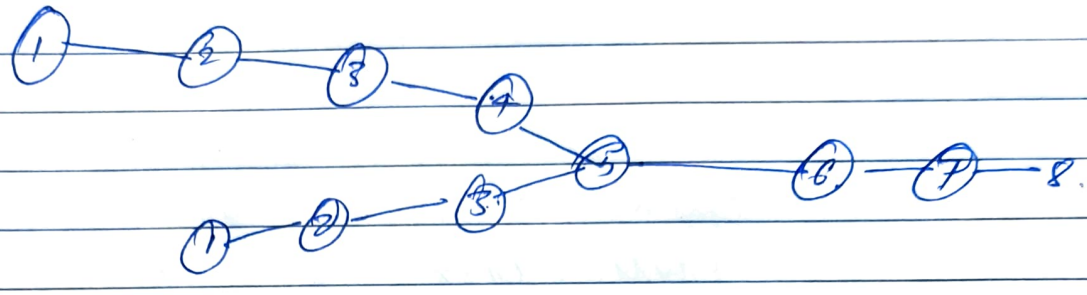
$ds = dxy$.

① cycle of length.
② rotation.
③ x.

APCO
Date :
Page (Educative.io system design)

160 Qu.

Find intersection point



8 - 7 = 1

smaller list में

61, 19, segregate odd, even.

LinkedList के

Note:- LinkedList के सारे fn को खुद लिखना होता है exception वहाँ लिखने से interviewer impress हो जाता है.

APCO
Date :
Page :

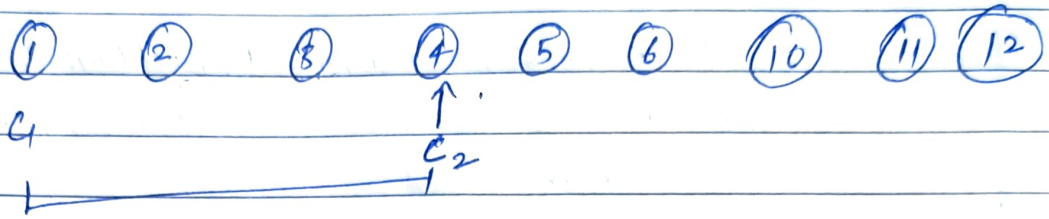
Code 11/100

```
ListNode* getIntersectionNode(ListNode *headA, ListNode *headB) {
    if(headA == nullptr || headB == nullptr){
        return nullptr;
    }
    ListNode* prev = headA;
    ListNode* curr = headA;
    while(curr != nullptr){
        prev = curr;
        curr = curr->next;
    }
    prev->next = headB;
    ListNode *ans = detectCycle(headA);
    prev->next = nullptr;
    return ans;
}
```

APCO
Date :
Page :

(11)

Q) Remove nth node from last. 4th node



⇒ make gap of n and run loop till C2 goes to end.

~~Edge~~ Case.

when.

APCO
Date :
Page :

Remove nth Node from end. // (19) Leetcode

```
ListNode* removeNthFromEnd(ListNode *head, int n){
    if(n==0 || head == nullptr){
        return head;
    }
    ListNode* c1 = head;
    ListNode* c2 = head;

    while(n-->0){
        c2= c2->next;
    }

    if(c2 == nullptr) return head->next;
    while(c2->next != nullptr){
        c1 = c1->next;
        c2 = c2->next;
    }
    c1->next = c1->next->next;
    return head;
}
```

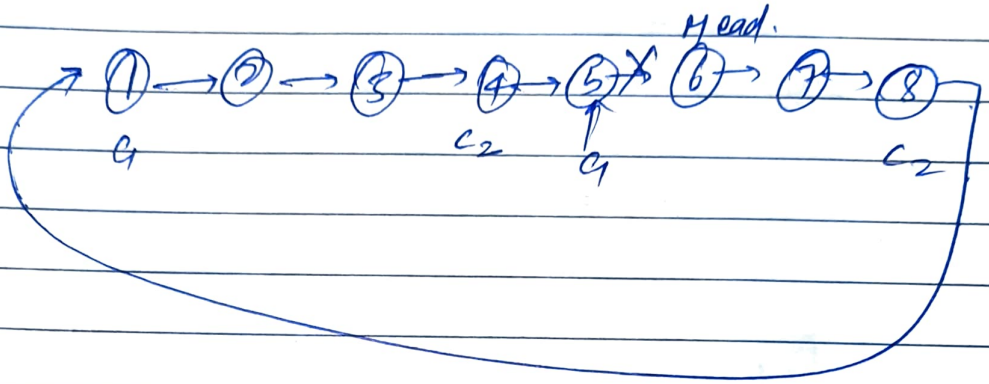
APCO
Date :
Page :

(61) Rotate ~~Left~~ Right

1 2 3 4 5

5 1 2 3 4

k Rotation बाएँ से आगे रख दो



APCO
Date :
Page :

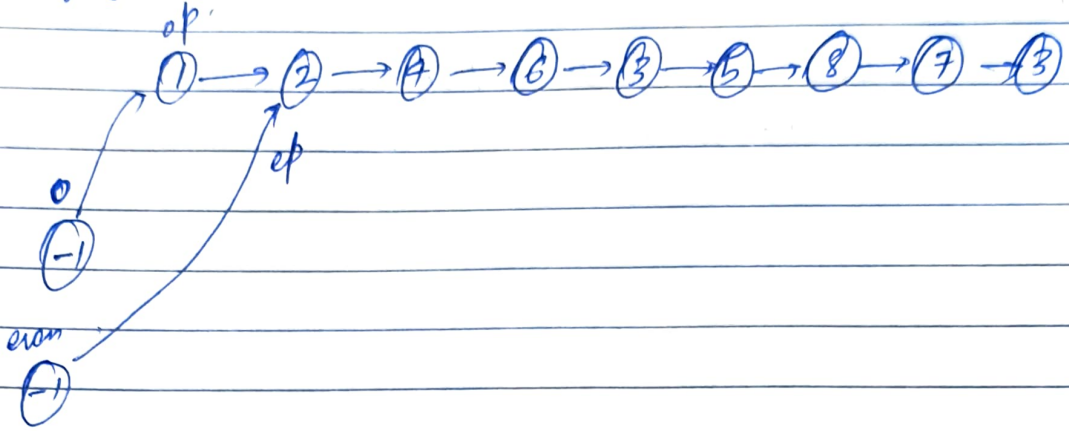
// Code 01 (Rotate Right).

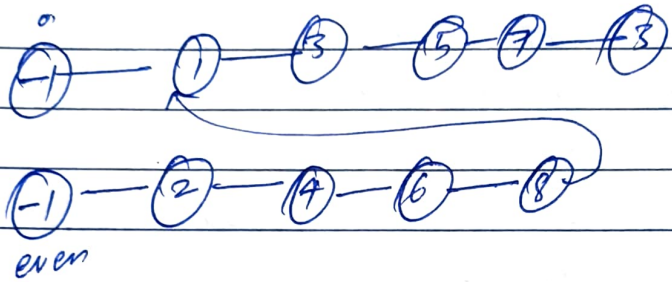
```
ListNode *rotateRight(ListNode *head, int k) {
    if(head == nullptr || head->next == nullptr ||
            k==0) {
        return head;
    }
    int len = 0;
    ListNode* curr = head;
    while(curr != nullptr) {
        curr = curr->next;
        len++;
    }
    k %= len;

    if (k==0) return head;
    ListNode* c1 = head;
    ListNode* c2 = head;

    while(k-->0)
        c2 = c2->next;

    while(c2->next != nullptr) {
        c1 = c1->next;
        c2 = c2->next;
    }
    c2->next = head;
    head = c1->next;
    c1->next = nullptr;
    return head;
}
```

APCO
Date :
Page :

segregate even and odd Numbers // geeksforgeeks

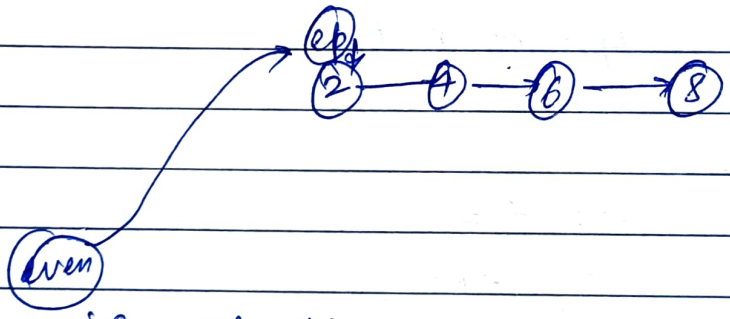
Q4.



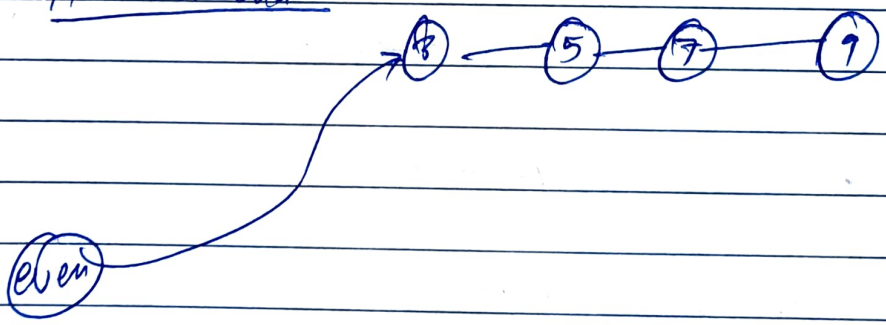


if all even:

U. dp tree



if all odd:



APCO
Date :
Page :

// Code geeks for geeks

```
ListNode* segregateOddEven(ListNode* head){
    if(head == nullptr || head->next == nullptr){
        return head;
    }
    ListNode* even = new ListNode(-1);
    ListNode* ep = even;

    ListNode* odd = new ListNode(-1);
    ListNode* op = odd;

    ListNode* curr = head;
    while(curr != nullptr){
        if(curr->val%2 == 0){
            ep->next = curr;
            ep = ep->next;
        }
        else{
            op->next = curr;
            op = op->next;
        }
        curr = curr->next;
    }
    ep->next = odd->next;

    return even->next;
}
```

APCO
Date :
Page :

142 // Linkedlist Cycle II.

```
ListNode* detectCycle(ListNode *head) {
    if(head == nullptr || head->next == nullptr){
        return nullptr;
    }
    ListNode* slow = head;
    ListNode* fast = head;

    while( fast != nullptr && fast->next != nullptr){
        slow = slow->next;
        fast = fast->next->next;
        if( slow == fast)
            break;
    }

    if( slow != fast) {
        return nullptr;
    }
    slow = head;
    while( slow != fast){
        slow = slow->next;
        fast = fast->next;
    }
    return slow;

}
```

APCO
Date :
Page :

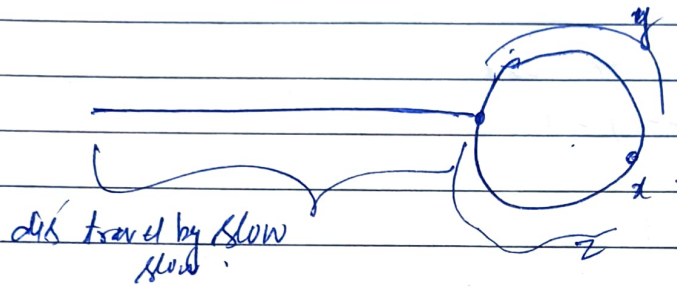
// Insight of y or y Calculation.

```
ListNode* detectCycle(ListNode* head){
    if(head == nullptr || head->next == nullptr){
        return nullptr;
    }
    ListNode* slow = head;
    ListNode* fast = head;

    int m=0, x=0, y=0, z=0;
    int distOfSlow=0;
    while(fast != nullptr && fast->next != nullptr){
        slow = slow->next;
        fast = fast->next->next;
        distOfSlow++;
        if(slow == fast)
            break;
    }
    if(slow != fast) return nullptr;

    slow = head;
    ListNode* node = fast;

    while(slow != fast){
        if(fast == node)
            m++;
        x++;
        slow = slow->next;
        fast = fast->next;
    }
```

APCO
Date :
Page :

```
    y = distOfSlow - x.
    return slow;
}
```

Here m is rotation need or rotation by fast pointer till slow reaches the cut point.



$x = z + (y + z)$ → rotation complete.

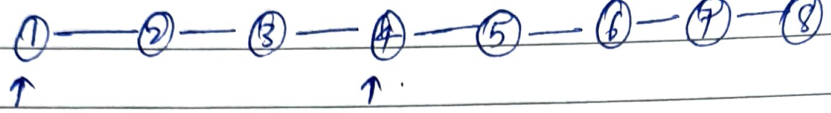
slow dist

fast pointer dist

extra Unit

APCO
Date :
Page :

Calculation on number line.

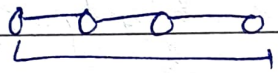


There are four types of diff.

(1) we generally do excluding 1 end $(4-1) = 3$.

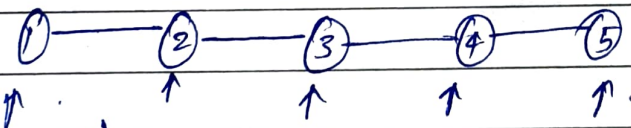
(2) excluding both end. $(4-1)-1 = 2$.

(3) including both end $= (4-1)+1$
$= 4$.



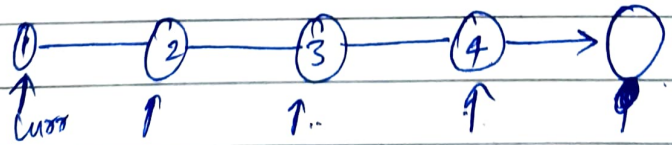
Note.

(4) 3 से 9 तक at any node the by going in while loop 'n' time i go 'n' step ahead from that ~~Rte~~ point.



move by 4

(5) If there are n node then doing 'n' iteration i will reach to the null or (n+1)th index.



APCO
Date :
Page :

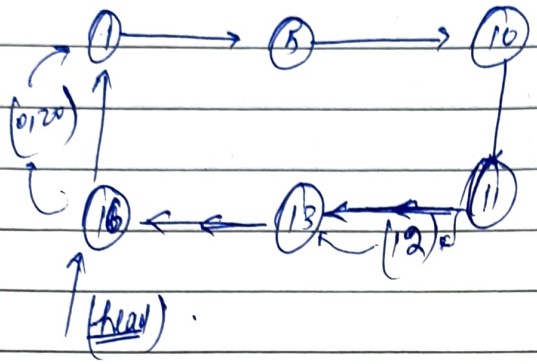
16/09/2020 - [Linked]

Agenda :-

1. (708) Insert into a Sorted Circular Linked List ||[599 LintCode]
2. (138) Copy List with random pointer.

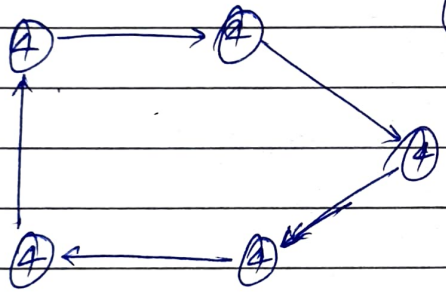
APCO
Date :
Page :

① 708 Insert into a sorted circular linked list

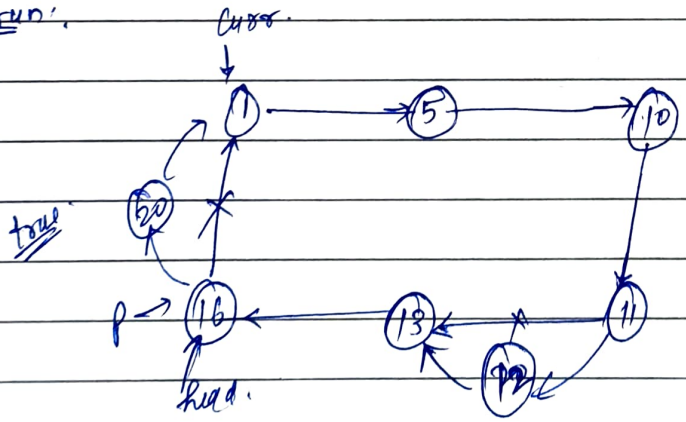


12/20/0

(5/3/4)



dry run:

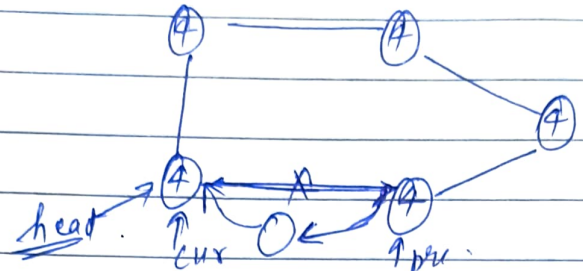


prev > next.

(val < curr).

APCO
Date :
Page :

(3/4/5).



if we go through whole list and do not find insertion point then come out insert p.

```
ListNode* insert(ListNode* head, int insertVal) {
    ListNode* node = new ListNode(insertVal);
    if(head == nullptr) {
        head->next = node;
        return node;
    }
    ListNode* prev = head;
    ListNode* curr = head->next;
    bool connect = false;
    while(curr != head) {
        if(insertVal >= prev->val && insertVal <= curr->val) {
            connect = true;
        }else if(prev->val > curr->val) {
            if(insert->val > prev->val || insert->val < curr->val
            {
                connect = true;
            }
        }
        if (connect)
            break;
        prev = curr;
        curr = curr->next;
    }
```

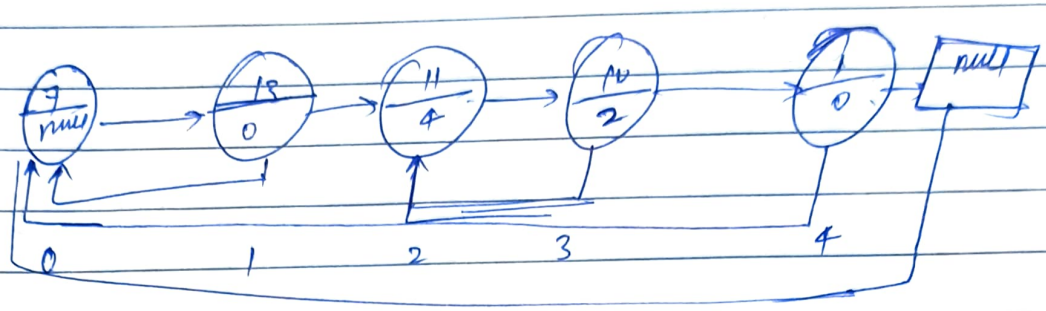
```
prev -> next = node;
node -> next = curr;

return head;
} // fn end
```

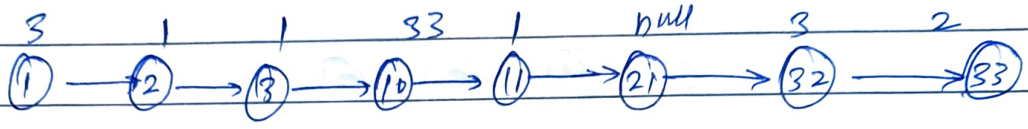
stack, hm, array, (string)

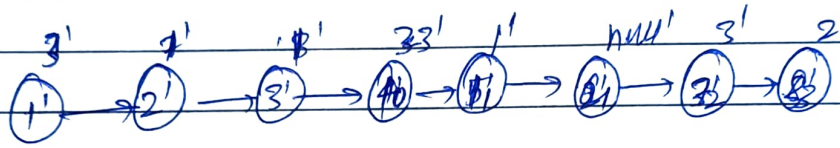
APCO
Date :
Page :

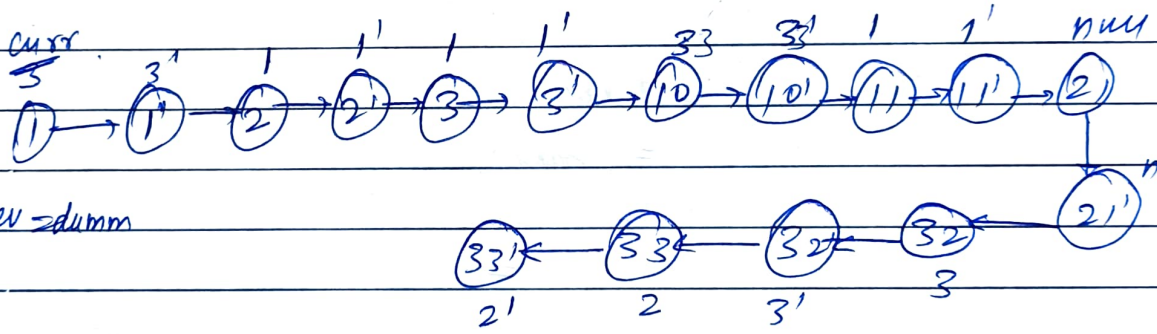
Very Imp.

Qu 138. Copy list with Random pointer.





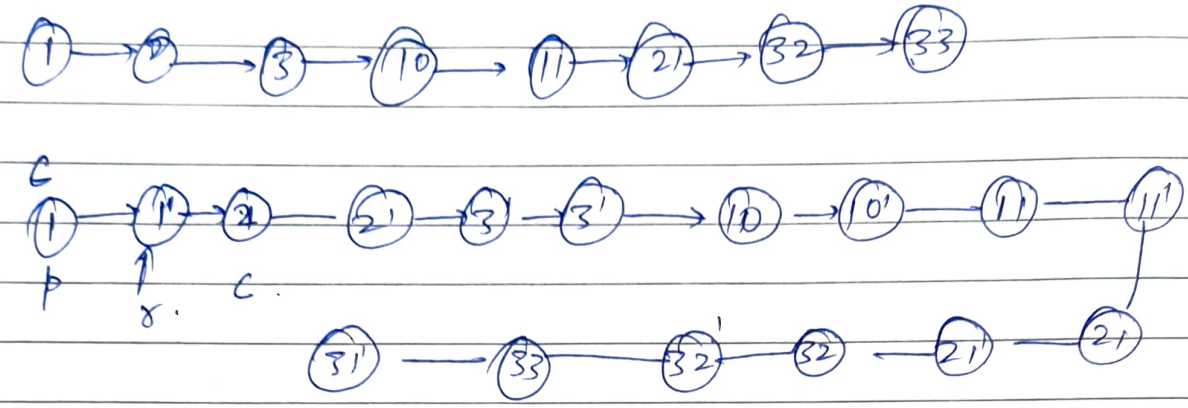


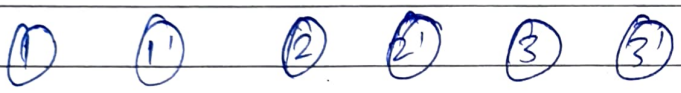


dummy = prev = dummy

T: O(3n) = O(n).

S: O(1) =





// 138

①
```
void copyNode (Node* head) {
    Node* curr = head;
    while (curr != nullptr) {
        Node* next = curr->next;
        Node* node = new Node (curr->val);
        curr->next = node;
        node->next = next;
        curr = next;
    }
}
```

APCO
Date :
Page :

② Eg

```
void copyRandomPointer(Node* head){
    Node* curr = head;
    while(curr != nullptr){
        if(curr->random != nullptr){
            curr->next->random = curr->random->ne
        }
        curr = curr->next->next;
    }
}
```

③ Eg

```
Node* extractMyll(Node* head){
    Node* dummy = new Node(-1);
    Node* prev = dummy;

    Node* curr = head;
    while(curr != nullptr){

        prev->next = curr->next; //link
        curr->next = curr->next->next;

        prev = prev->next;   //move
        curr = curr->next;
    }

    return dummy->next;
}
```

APCO
Date :
Page :

(4)

```
Node* copyRandomList (Node* head) {
    if (head == nullptr) {
        return head;
    }

    CopyNode (head);                    // O(n)
    copyRandomPointers (head);          // O(n)
    return extractMyLL (head);          // O(n).
}
```