Recursion :-

function, calling it self.

2 imp. things on Recursion. i.e;

faith

Expection

* print of factorial numbers

faith

$n = 5$

Expection

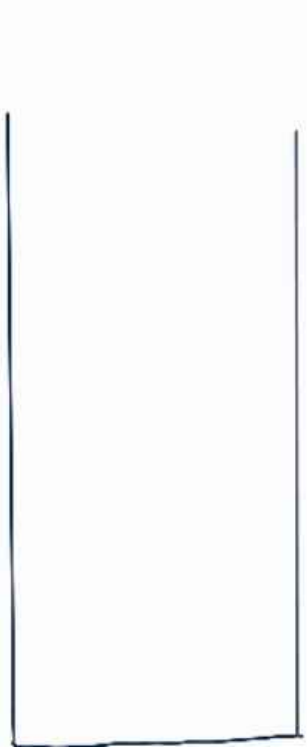
(5 * (4 x 3 x 2 x 1))

5 x 4 x 3 x 2 x 1

n x fact(n-1)

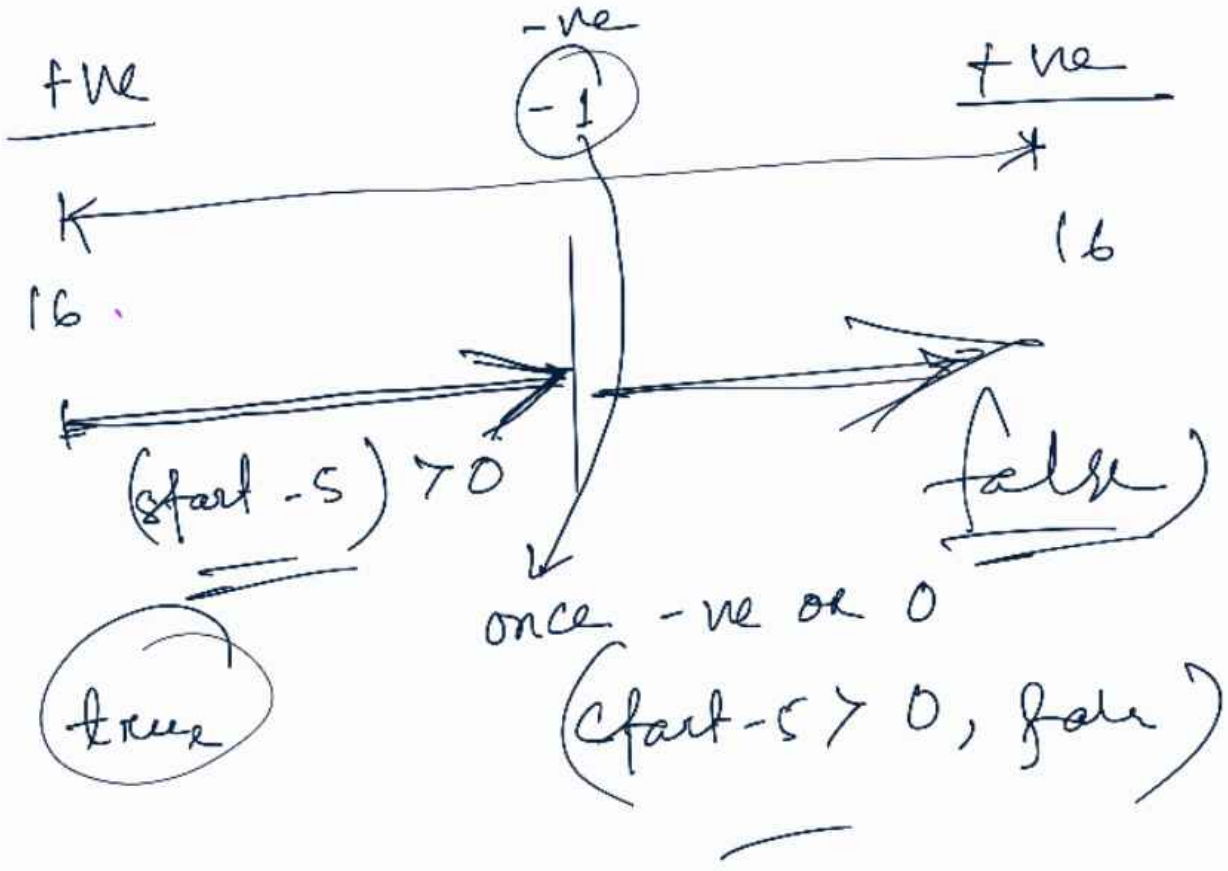
* Print Pattern

$n = 16$



O/P

16



Q. Find the first index of target element.

```
int firstIndex(int idx, int[] arr, int target){

    if(idx == arr.length) return -1;

    if(arr[idx] == target){

        return idx;

    return firstIndex(int idx+1, 3 = target
                      arr, target);
}
```

7	3	8	3

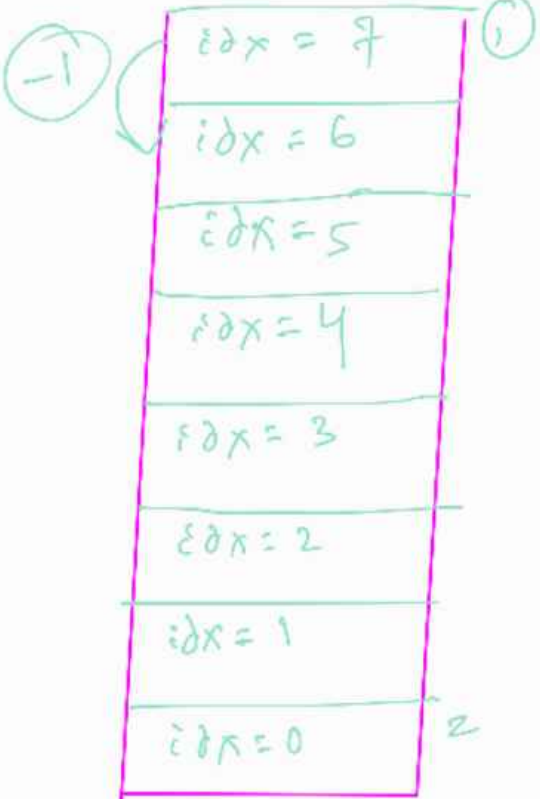
faith : (idx+1 → n)

Exp : (idx → n)

Q. Find the last index of target element.

```
int LastIndex(int idx, int[] arr, int target) {
    ① if (idx == arr.length) return -1;
    ② int sAns = LastIndex(idx+1, arr, target);
       int ans = -1;
    ③ if(sAns == -1) {
           if(arr[idx] == tar) {
               ans = idx;
           else  ans = sAns;
       }
       else ans = sAns;
}
```

0	1	2	3	4	5	6
7	3	2	2	3	0	1

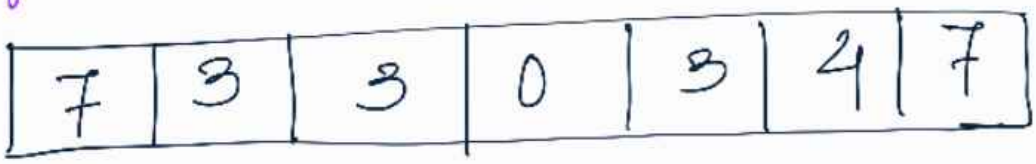


sAns = -1

ans = -1

Q. Return an array containing indices of the target element.

arr ⇒



target ⇒ 3

```
int getIdx(int idx, int[] arr, int tar, int count){
    if(idx == arr.length){
        int[] ans = new int[count];
        return ba;
    }
    int[] ans;
    if(arr[idx] == tar){
        return(idx+1, arr, tar, count+1);
    }
    else return(idx+1, arr, tar, count);

    if(arr[idx] == tar){
        ans[count] = idx;
    }
}
```

Q. Subsequences of String using recursion?

```
ArrayList<String> getSubSequences(String str) {
    if(str.length() == 0) {
        AL<String> ba = new AL<>();
        ba.add(" ");
        return ba;
    }

    char first_char = str.charAt(0);
    String ss = str.substring(1);

    ArrayList<String> sAns = getSubSequences(ss);
    ArrayList<String> ans = new ArrayList<>();

    for(int i=0 → sAns.size() ) {
        String sub = sAns.get(i);
        ans.add(sub);
    }

    for(int i=0 → sAns.size()) {
        String sub = sAns.get(i);
        String ansSub = first_char + sub;
        ans.add(ansSub);
    }

    return ans;
}
```

Q. N^{th} even fibonacci number.

```
int evenfib(int n) {
    if (n==0) return 0;
    if(n==1) return 2;
    return 4*evenfib(n-1)+evenfib(n-2);
}
```

0 1 1 (2) 3 5 8 13 21 34 55 89

= 0 = 1 = 2 = 3 = 4

n=2

$4\,eve(n-1) + eve(n-2)$

$= 4 \times 2 + 0$

$= 8 + 0 = 0$

n=3

$4 \times eve(n-1) + eve(n-2)$

$= 4\,eve(3-1) + eve(3-2)$

$= 4 \times eve(2) + eve(1) \rightarrow 2$

$= 4\{4 \times (eve(2-1) + eve(2-2)\} + 2$

$= 4(4 \times 2 + 0) + 2$

$= 4 \times 8 + 2 = 32 + 2 = 34$

144 = 4

2

Explanation:

(n-6) (n-5) (n-4) (n-3) (n-2) (n-1) n

0 1 1 2 3 5 8 13 21 34 55 89

= = = = =

0 1 2 3 144

4

for, f(0) = 0

f(1) = 1

f(2) = even f(n-1) + even f(n-2)

should be even should be even

$= f(n-1) + f(n-2)$

but I need (n-3) & (n-6) for prev even fib number to get f(n).

$= [f(n-2) + f(n-3)] + [f(n-3) + f(n-4)]$

$= 2f(n-3) + [f(n-2) + f(n-4)]$

$= 2f(n-3) + [f(n-3) + f(n-4)] + f(n-4)$

$= 3f(n-3) + 2f(n-4)$

$= 3f(n-3) + f(n-4) + f(n-4)$

$= 3f(n-3) + f(n-4) + [f(n-5) + f(n-6)]$

$= 3f(n-3)] + [f(n-4) + f(n-5)] + f(n-6]$

$= [3f(n-3)] + f(n-3)] + f(n-6)$

$\Rightarrow = 4f(n-3) + f(n-6)$

prev even(n-1) prev even(n-2)

Q. Robbing house :-

```
int robHouse(int idx, int end, int[] arr) {
    if (idx > end) return 0;
    int rob = robHouse(idx+2, end, arr) +
                                arr[idx];
    int skip = robHouse(idx+1, end, arr);
    return Math.max(rob + skip);
}

startFrom0 = robHouse(0, n-2, arr);
startFrom1 = robHouse(1, n-1, arr);
sop(Math.max(startFrom0, startFrom1));
```

CodingBat_Recursion :-

1) factorial

```
int fact(int n) {
    if(n==1) return 1;
    return n * fact(n-1);
}
```

2) Bunny Ears

```
int bunnyEars(int n) {
    if (n==0) return 0;
    return 2 + bunnyEars(n-1);
}
```

3) fibonacci

```
int fib(int n){
  if(n==0) return 0;
  if(n==1) return 1;
  return f(n-1)+f(n-2);
}
```

4) bunnyEars2

```
int bunnyEars2 {
  if(n==0) return 0;
  if(n%2==0)
    return 3+bunnyEars2(n-1)
  return 2+bunnyEars2(n-1)
}
```

5) triangle

```
int triangle(int n) {
  if(row==0) return 0;
  return row+triangle(n-1);
}
```

5) sumDigits

```
int sumDigits(int n) {
  if(n==0) return 0;
  return (n%10)+
      sumDigits(n/10);
}
```

6) countF

```
int countF(int n) {
  if(n==0) return 0;
  if(n%10==7)
    return 1+countF(n/10);
  return countF(n/10);
}
```

7) count8

```
int count8(int n) {
  if(n==0) return 0;
  if(n%10==8)
    return 1+count8(n/10);
  return count8(n/10);
}
```

8) powerN

```
int powerN(int base, int n){
  if(n==0) return 1;
  return base*powerN(base,n-1);
}
```

9) CountX

```
int CountX(String str) {
  if (str.length() == 0) return 0;
  if(str.charAt(0) == 'x')
        return 1 + CountX(str.substring(1));
  return CountX(str.substring(1));
}
```

10) CountHi

```
int countHi(String str){
  if(str.length() < 2)
        return 0;
  if(str.substring(0,2).equals("hi"){
        return 1 + countHi(str.substring(1));
  return CountHi(str.substring(1));
}
```

11) ChangeXY

```
String changeXY(String str){
  if(str.Length == 0)
        return "";
  if(str.charAt(0) == 'x') {
        return 'y' + changeXY(str.substring(1));
  return str.charAt(0) + changeXY(str.sub(1));
}
```

12) changePi

```
String changePi (String str) {
    if (str.length() == 0)
        return "";
    if(str.substring(0,2).equals("pi") && str.length() >= 2) {
        return "3.14" + changePi(str.sub(1));
    }
    return str.charAt(0) + changePi(str.sub(1));
}
```

13) nox

```
String nox(String x) {
    if (str.length() == 0)
        return "";
    if(str.charAt(0) == 'x')
        return nox(str.sub(1));
    return str.charAt(0) +
        nox(str.sub(1));
}
```

14) array6

```
boolean array6 (int[] nums, int index) {
    if (index == nums.length) return false;
    if( nums[index] == 6 ) return true;
    return array6(nums, index+1);
}
```

15) array11

```
int array11(int[] nums, int index) {
    if(index == nums.length)
        return 0;
    if(nums[index] == 11)
        return 1 + array11(nums, index+1);
    return array11(nums, index+1);
}
```

16) array220

```
boolean array220(int[] nums, int index) {
    if(index >= nums.length-1)
        return false;
    if(nums[index] * 10 == nums[index+1])
        return true;
    return array220(nums, index+1);
}
```

17) allStar

```
String allStar(String str) {
    if(str.length() == 0) return str;
    return str.charAt(0) + "*" +
                allStar(str.substring(1));
}
```

18) pairStar

```
String pairStar (String str) {
    if (str.length() < 2) return str;
    if(str.charAT(0) == str.charAt(1))
        return str.charAt(0) + "*" +
                        pairStar(str.substring(1));
    return st.charAt(0) + pairStar(str.substring(1));
}
```

19) endX

```
String endX (String str) {
    if(str.length() == 0) return str;

    if( str.charAt(0) == 'x')
        return endX (str.substring(1)) +
                        Str.charAt(0);

    return str.charAt(0) + endX (str.sub(1));
}
```

20) CountPairs

```
int CountPairs(String str) {
    if(str.length() <= 2) return 0;
    if(str.charAt(0) == str.charAt(2))
        return 1 + countPairs(str.subStr(1));
    return countPairs(Str.substr(1));
}
```

21) CountAbc

```
int CountPairs(String str) {
    if(str.length() <= 2) return 0;
    if(str.substring(0,3).equals("abc") || ("aba"))
        return 1 + countPairs(str.substr(1));
    return countPairs(str.substr(1));
}
```

22) Count11

```
int count11(String str) {
    if (str.length() < 2) return 0;
    if (str.substring(0,2).equals("11")
        return 1 + count11(str.sub(2));
    return count11(str.sub(1));
}
```

23) stringClean

```
String stringClean(String str) {
    if(str.length() < 2) return str;
    if(str.charAt(0) == str.charAt(1))
        return stringClean(str.sub(1));
    return str.charAt(0) + stringClean
                (str.sub(1));
}
```

24) CountHi2

```
int countHi2(String str) {
    if (str.length() < 2) return 0;
    if(str.charAt(0) != 'x' &&
        str.substring(1,3).equals("hi")) {
        return 1+countHi2(str.substring(2));
    return countHi2(str.sub(1));
}
```

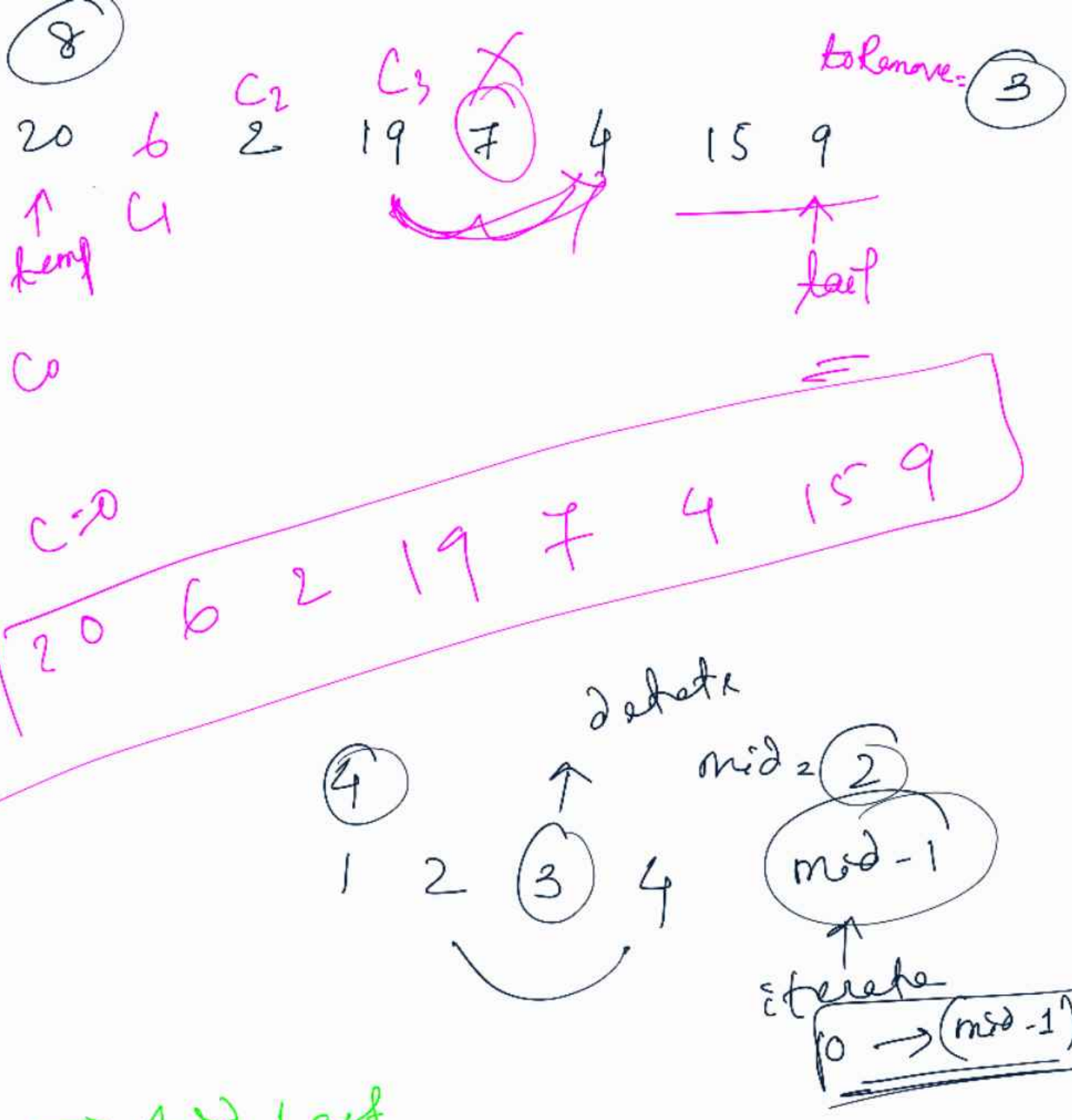
LinkedList :-

```
Node nn = new Node(val);
if(head == null)
    head = tail = nn;
tail.next = nn;
tail = nn;
```

Add Last

```
Node nn = new Node(val);
if(head == null) head == tail = nn;
nn.next = head;
head = nn;
```

Add first



→ Add Last

→ Add First

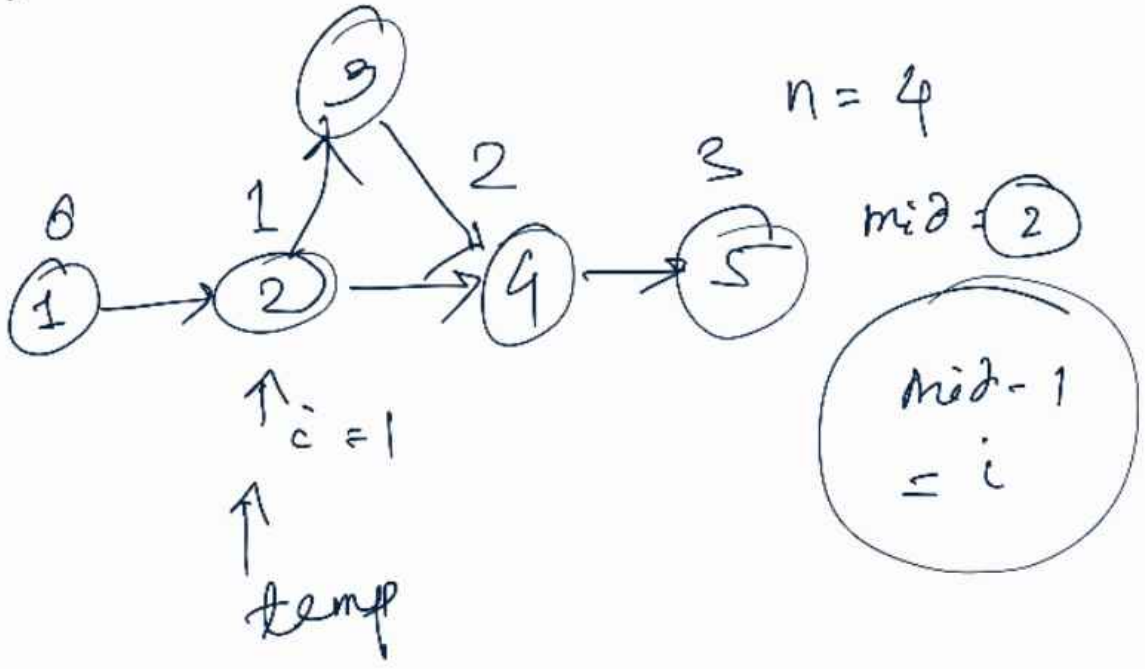
→ Remove First || Last

→ get Node at given position

→ Add a node at given index.

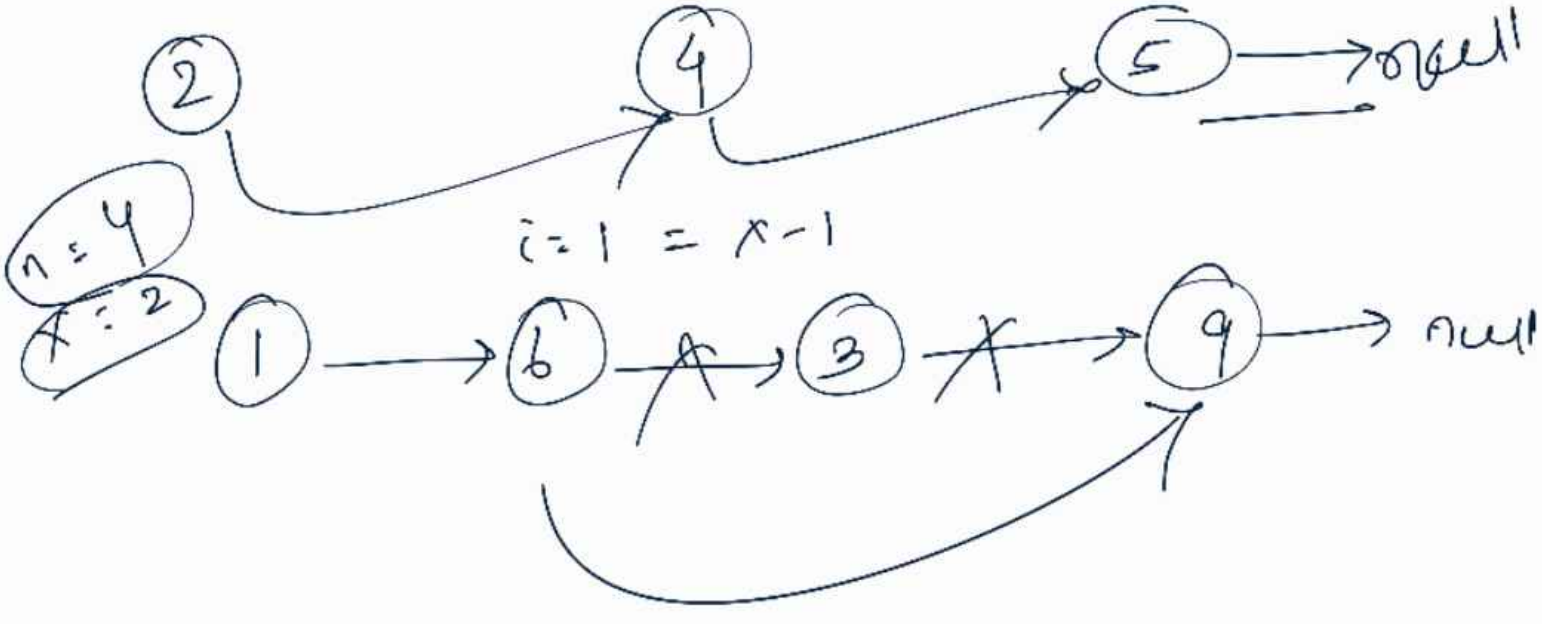
→

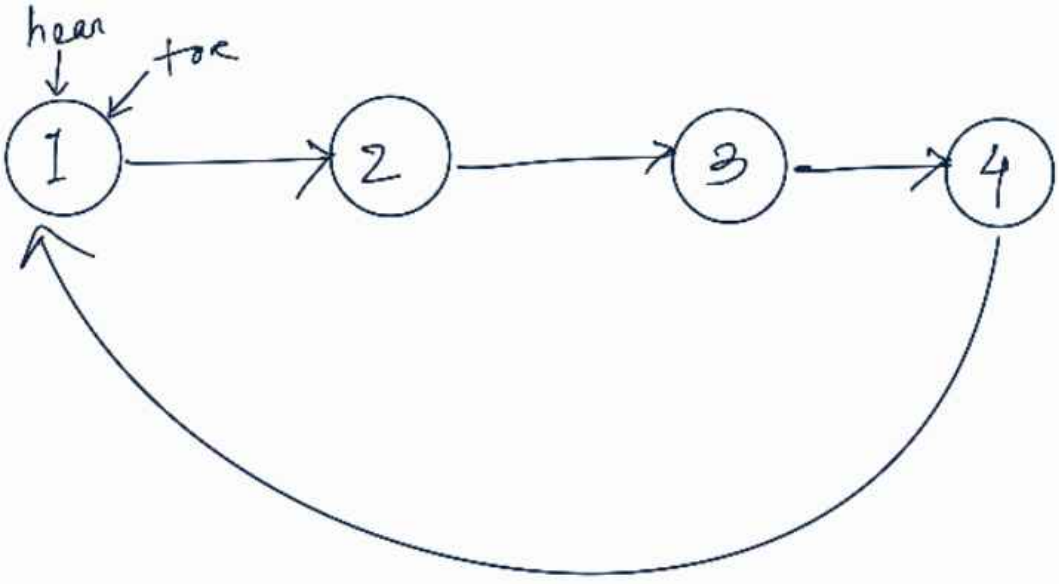
```
for(int i=0 → nodes ; i++) {
    if(mid == i) {
        nn.next = temp.next;
        temp.next = nn;
    }
}
```

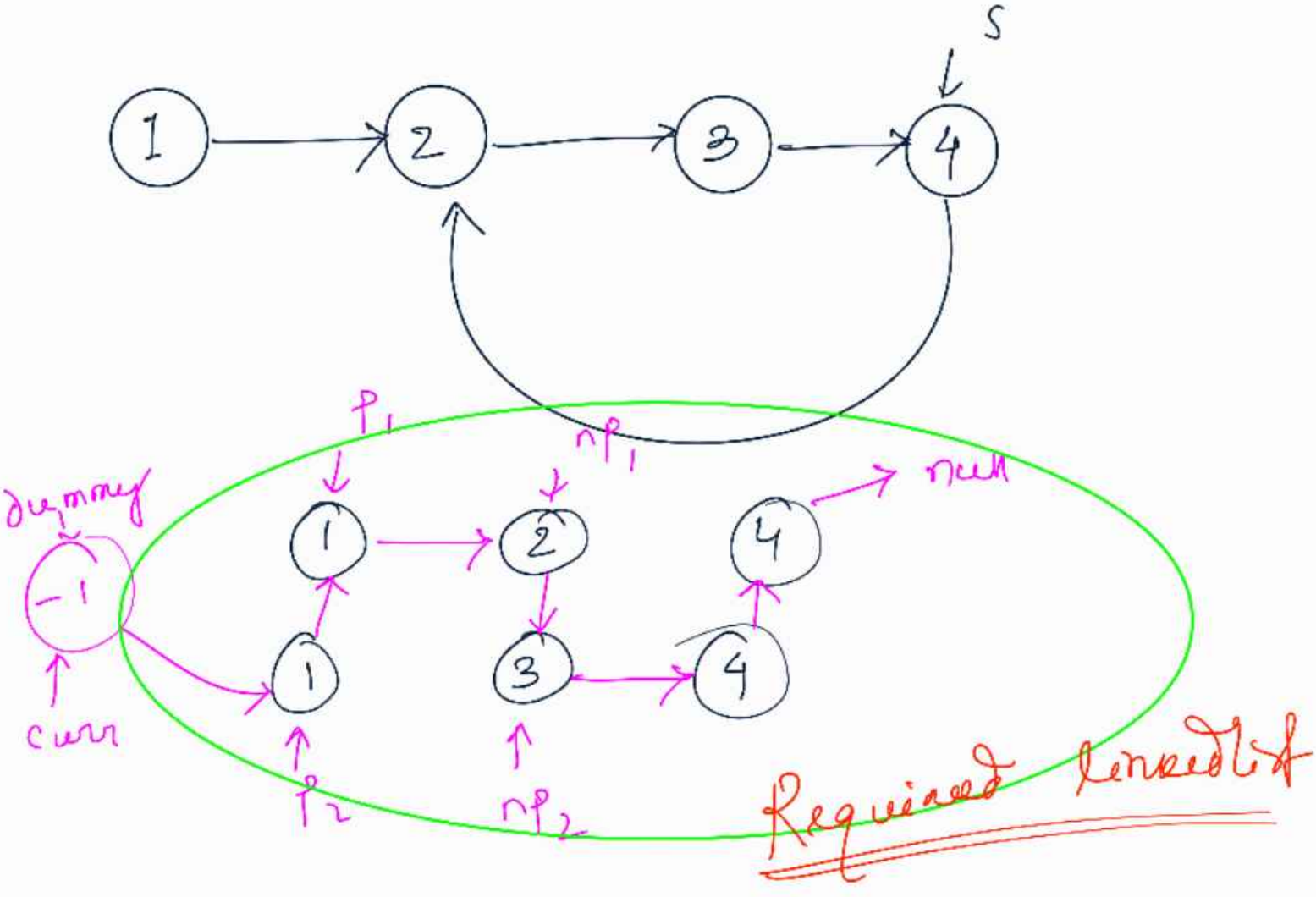


---

n = 3, x = 3 for(i=0 → 5)




head
toe
1
2
3
4

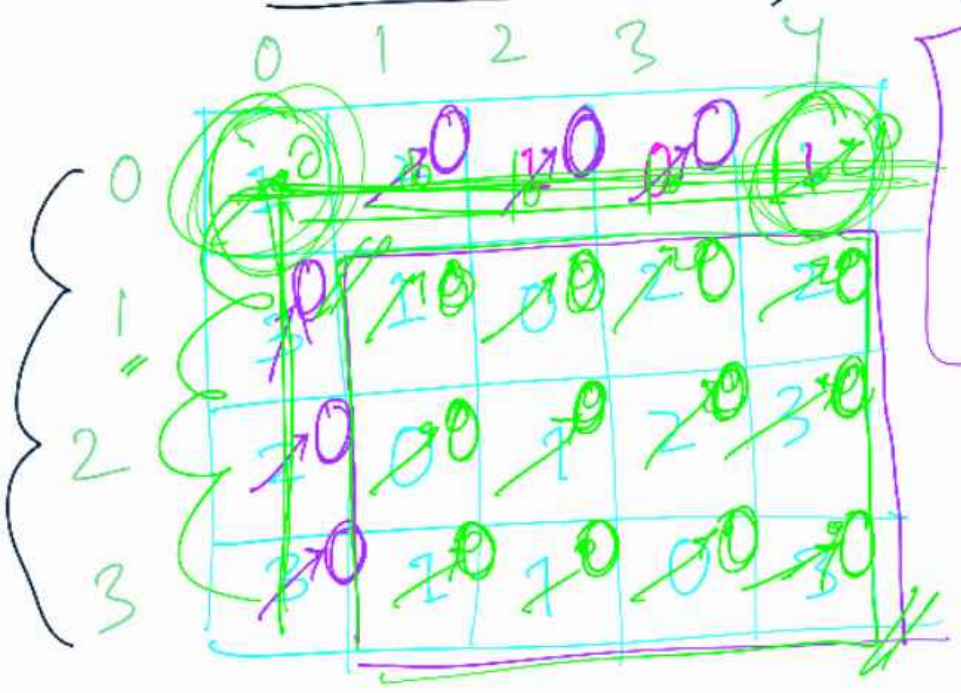

S
1
2
3
4
p_1
np_1
dummy
-1
curr
1
2
4
null
1
3
4
p_2
np_2
Required Linked list

```
void SetMatrixZeros(int[][] matrix) {
    Boolean isZeroCol = false;
    Boolean isZeroRow = false;
    int row = mat.length;
    int col = mat[0].length;
// check for first col
for(int i=0; i<row; i++) {
        if(mat[i][0] == 0) {
            isZeroCol = true;
            break;
        }
}
// for first Row
for(int i=0; i<col; i++) {
    if(mat[0][i] == 0) {
        isZeroRow = true;
        break;
    }
}
// Check except the 1st row & col
for(int i=1; i<row; i++) {
    for(int j=1; i<col; i++) {
        if(mat[i][j] == 0) {
            mat[i][0] == 0;
            mat[0][j] == 0;
}   }   }
```

1st col → for(int i=0 → row) {
(mat[i][0]) isZeroCol = true;
}

isZeroCol = true
isZeroRow = false

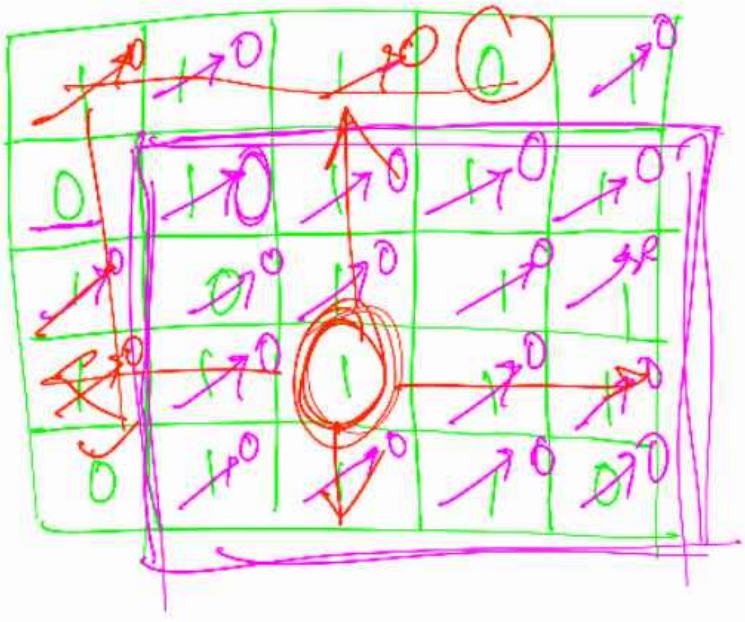
1st row → for(int i=0 →) {
(mat[0][0]) isZeroRow = true;
}



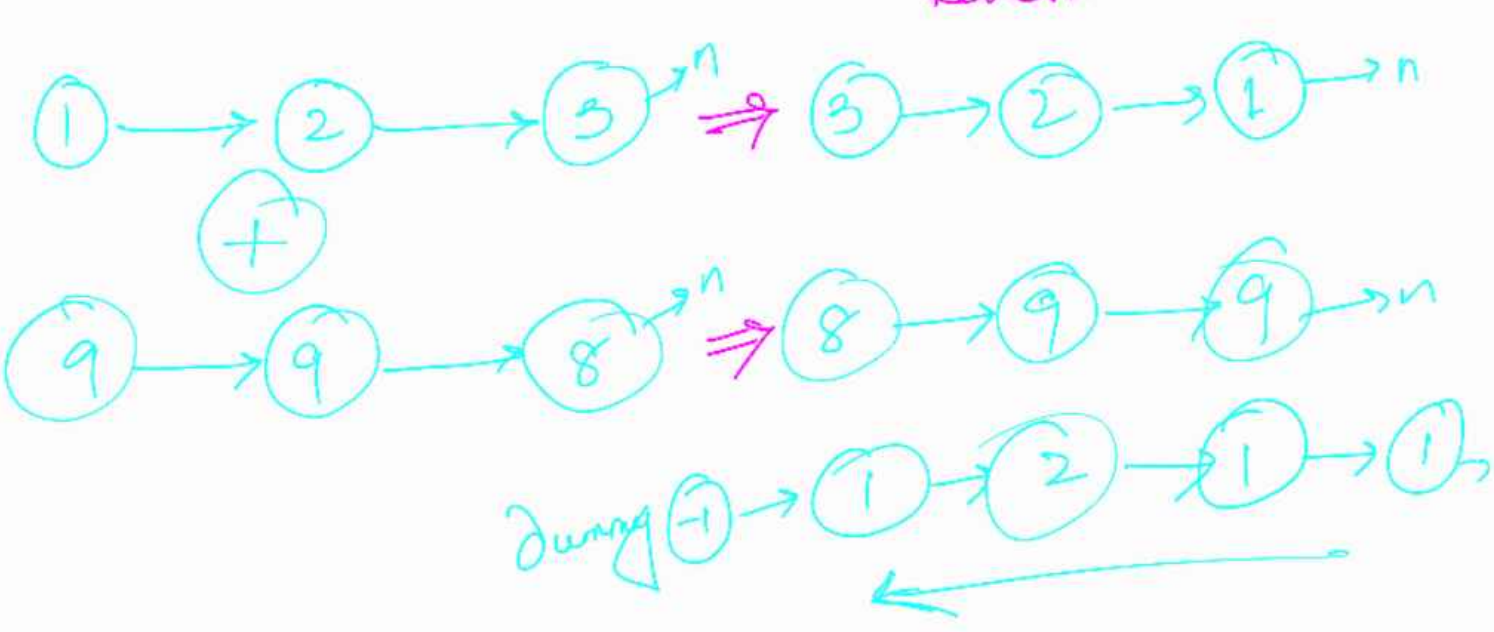
for(int i → row) {
for(int j → col) {
if(mat[i][j] == 0)
mat[i][0] = 0;
mat[0][j] = 0;
}
}

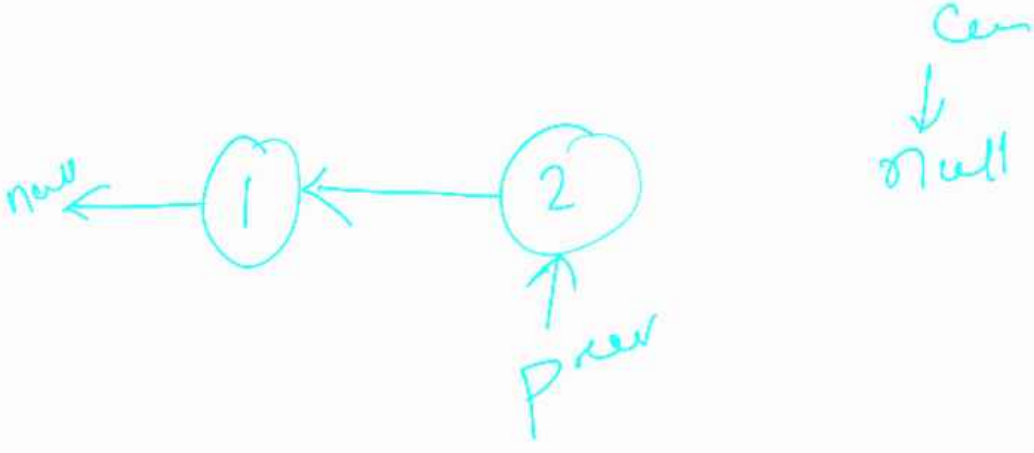
for(int i=1 → row) {
for(int j=1 → col) {
if(mat[i][0] == 0 ||
mat[0][j] == 0)
mat[i][j] = 0;
}
}

if(zeroCol) {
for(int i=0; i<col; i++){
mat[0][i] = 0;
}
}



Q. Add 1 to a number Representing linkedlist
or Add 2 nos. representing as linkedlist.





(5)

[1, 3, 5, 2, 2]

↑

rSum = 13 − 1 = 12 − 3 = 9 − 5 = (4)

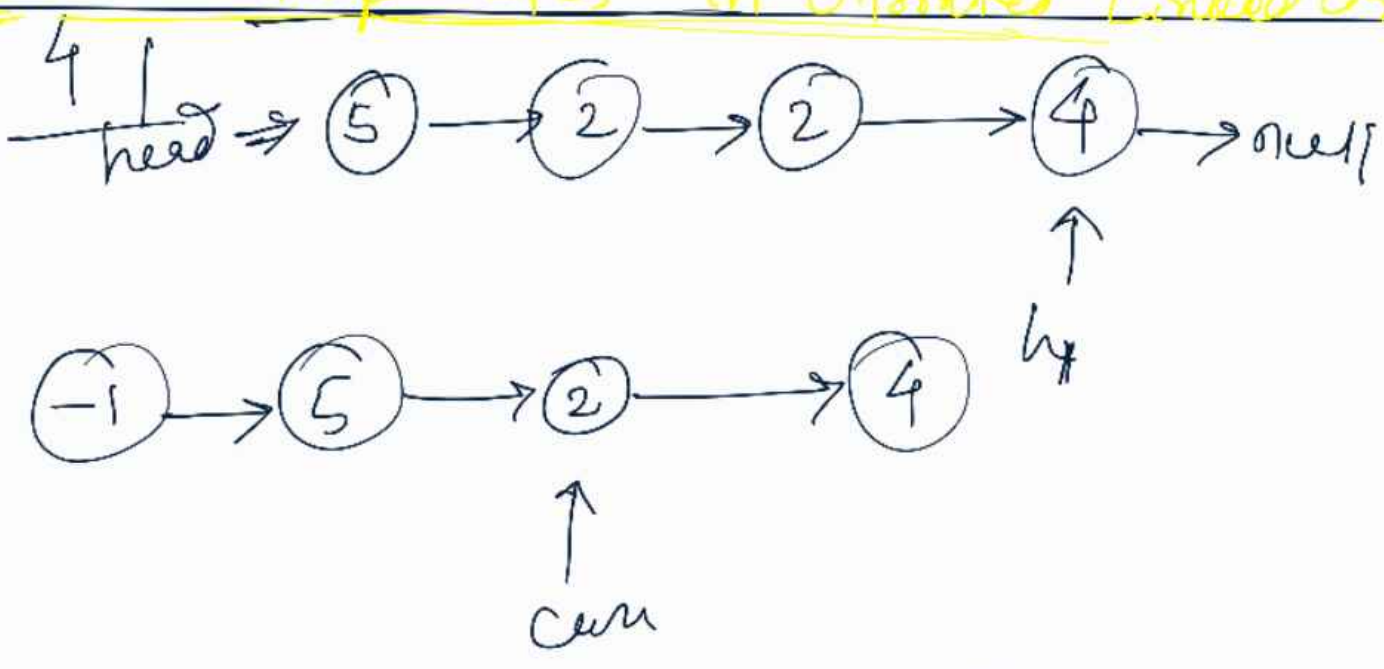
lSum = 0 + 1 = 1 + 3 = (4)

[1, 4, 9, 11, 13]

↑ ↑

$a[i] + a[i-1] = a[n-1]$

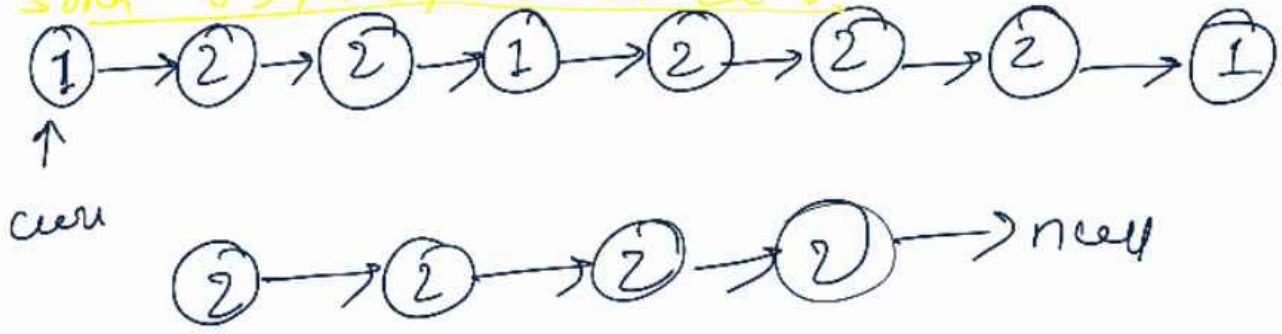
Q. Remove Duplicates in Unsorted Linked List

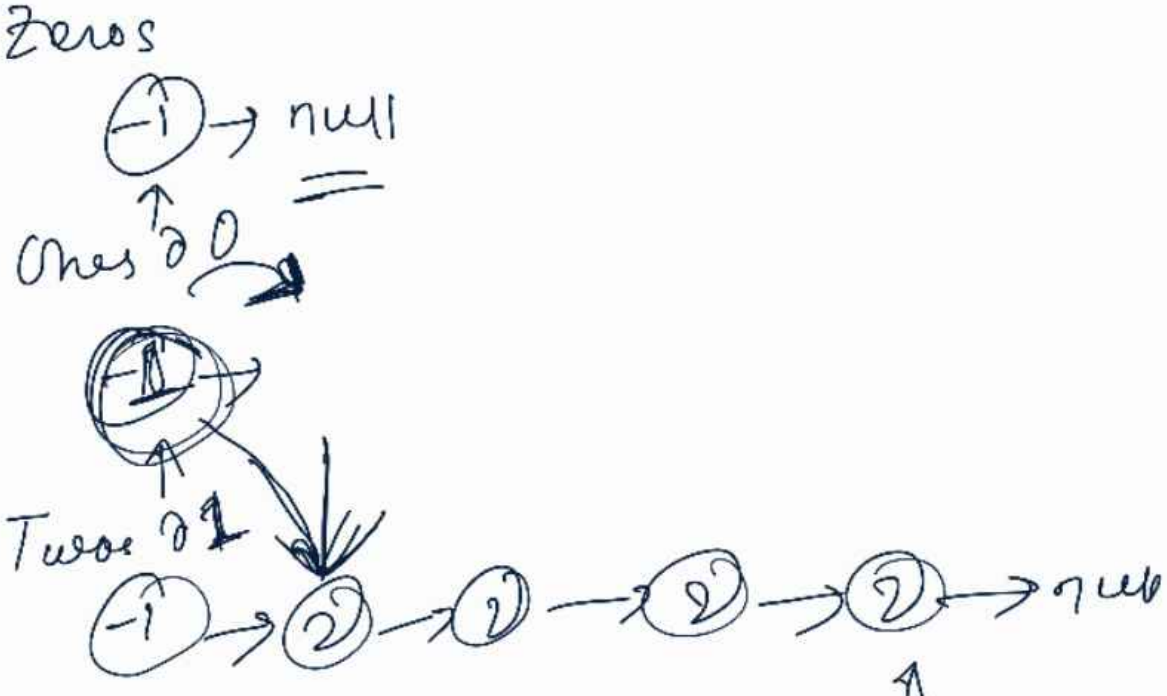


{ 5 : 1;
2 : 1;

}

Q Sort 0's, 1's, 2's on LL :-





A B C A C B A

=

i = 0

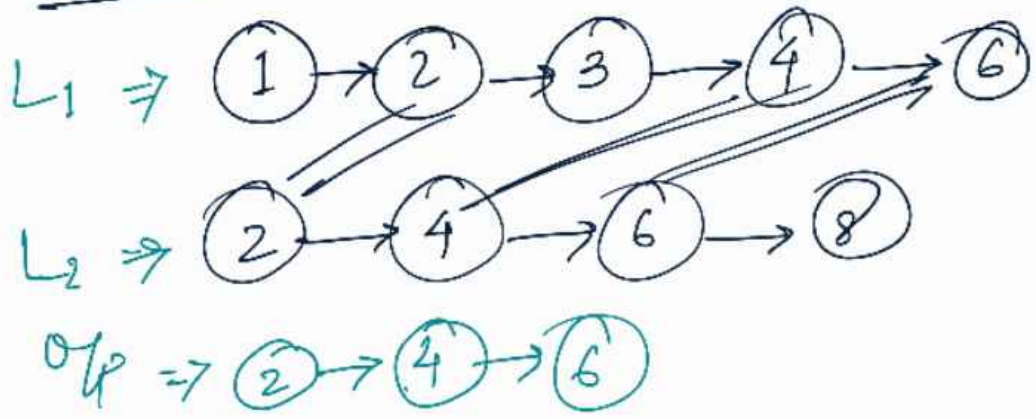
j = 4 - 1 = 3

while(0 < 3)

count = 0 1 2 3

Q. Intersection of Two sorted LinkedList.

Case - 1

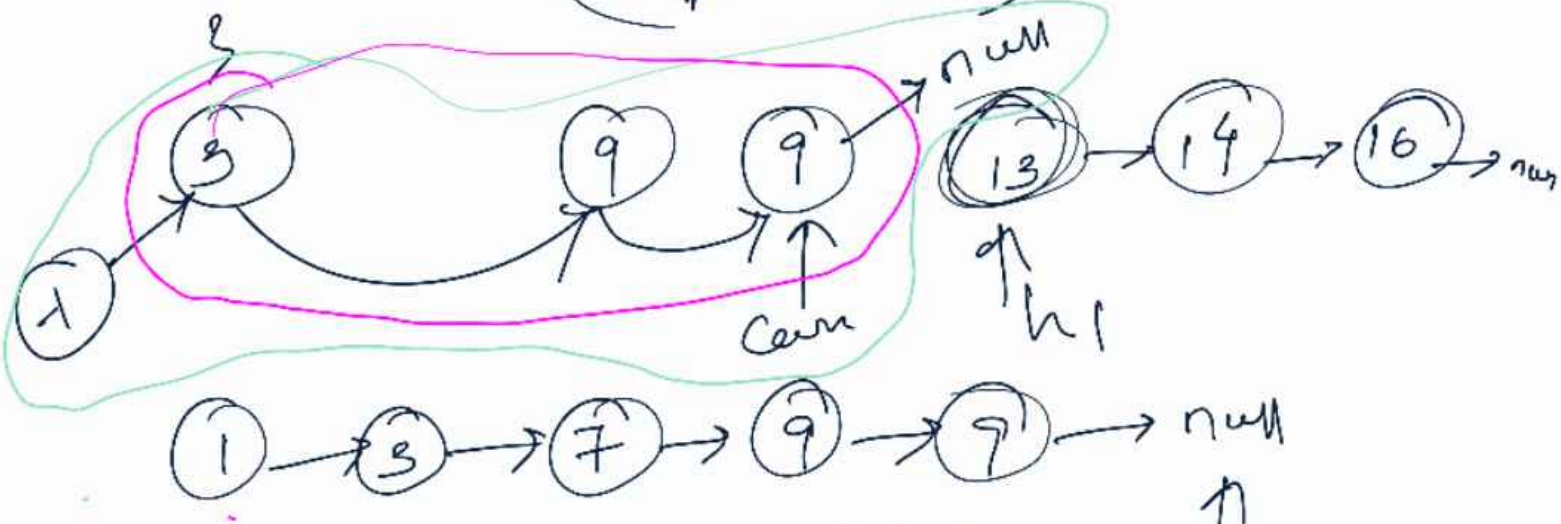


check (l_1.data == l_2.data)

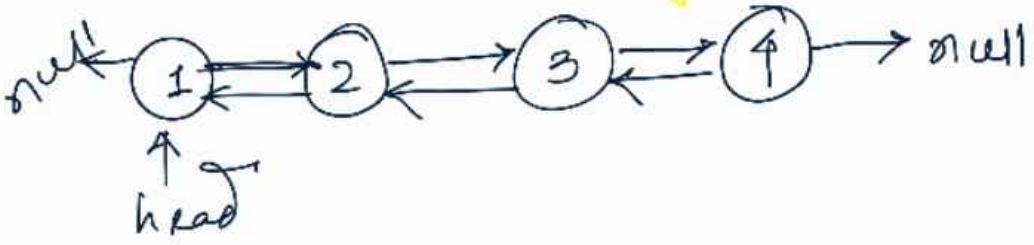
else if (l_1.data < l_2.data)

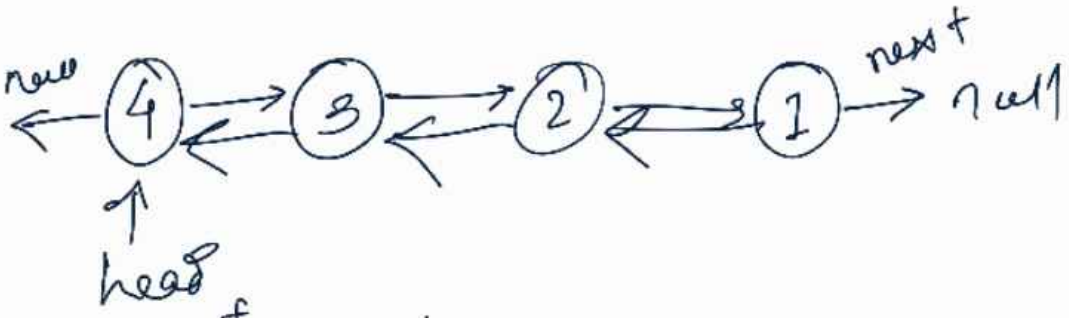
(shift >> l_1)

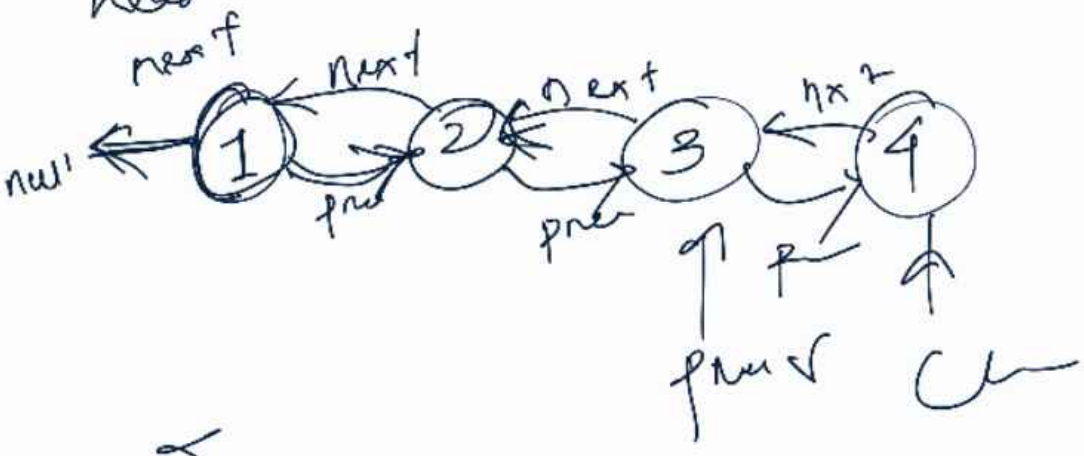
else (shift >> l_2)

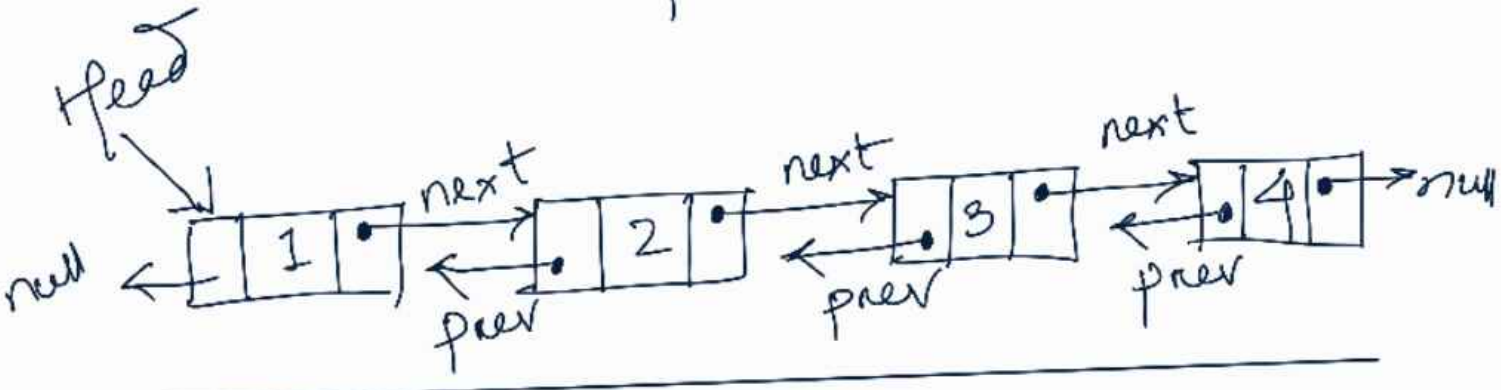


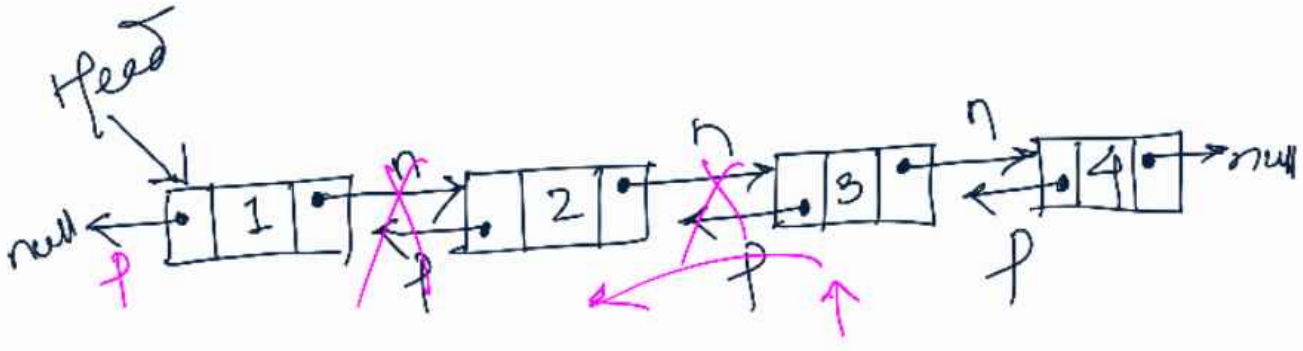
Q Reverse a Doubly LinkedList :—







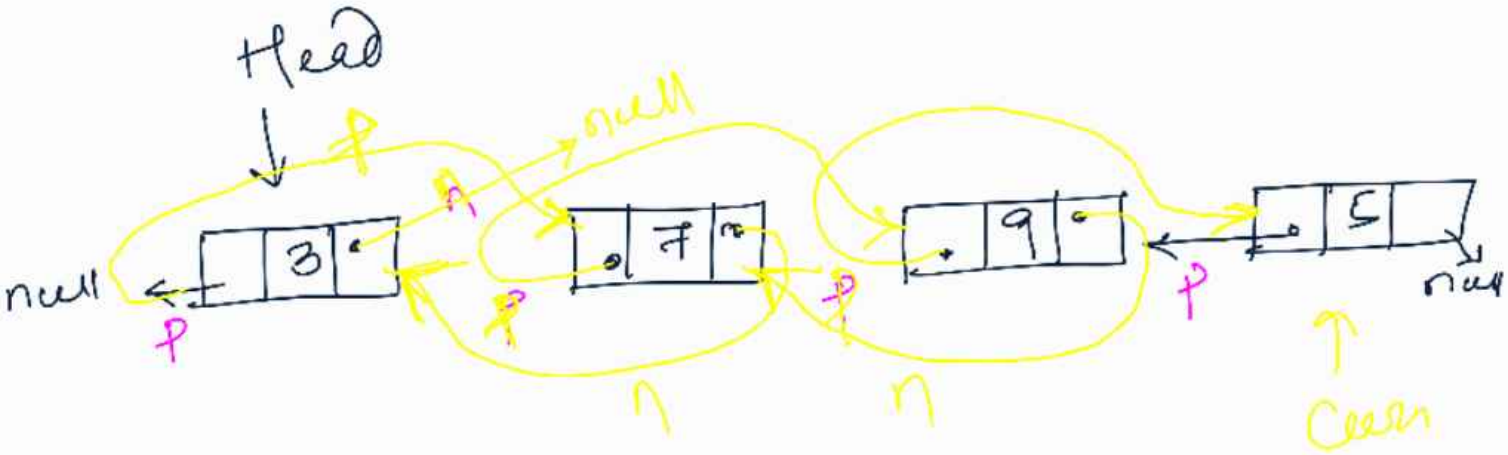




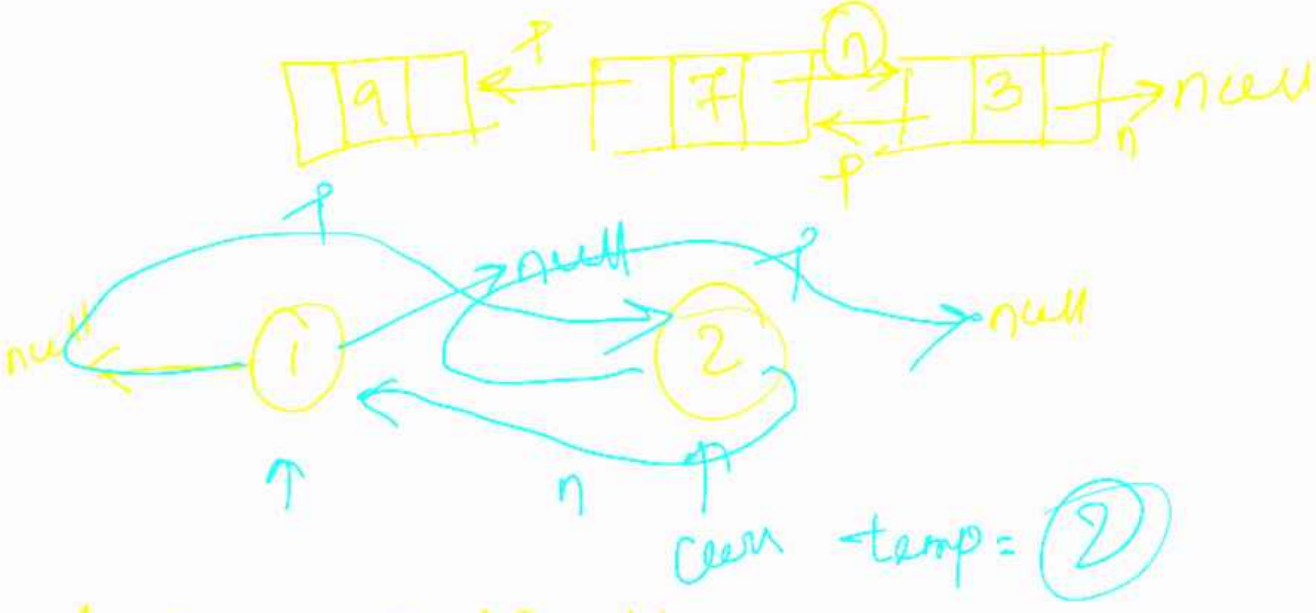
temp = ~~null~~ 1 2 3

```
while (curr != null) {
    ① temp = curr.prev;
    ② curr.prev = curr.next;
    ③ curr.next = temp;
    ④ temp = curr;
    ⑤ curr = curr.prev;
}
```

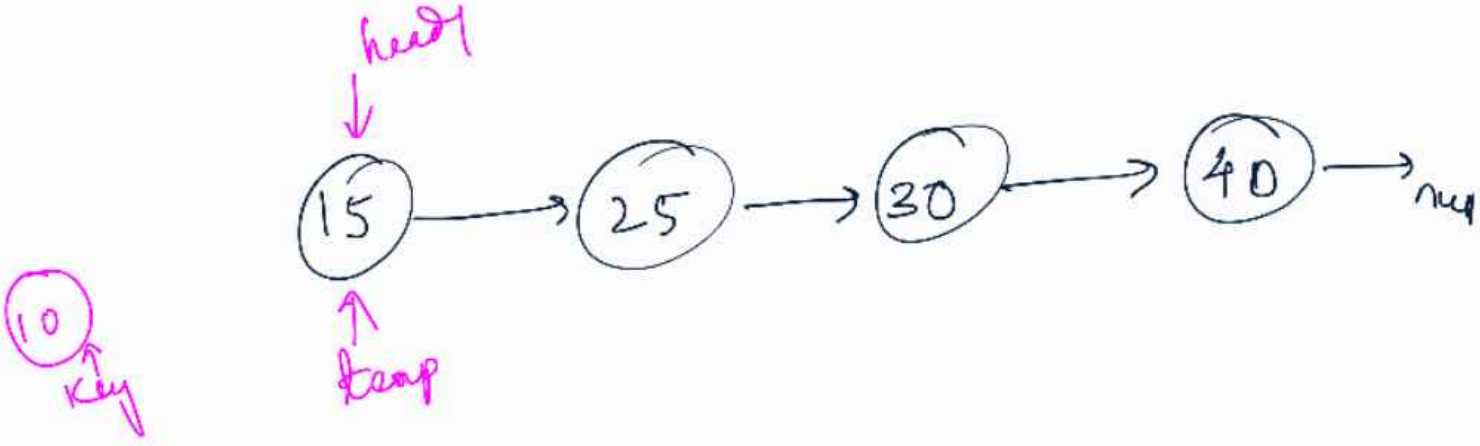
not



temp: null null 3 7 9



Q. Insert In a Sorted LL :-



Case-1

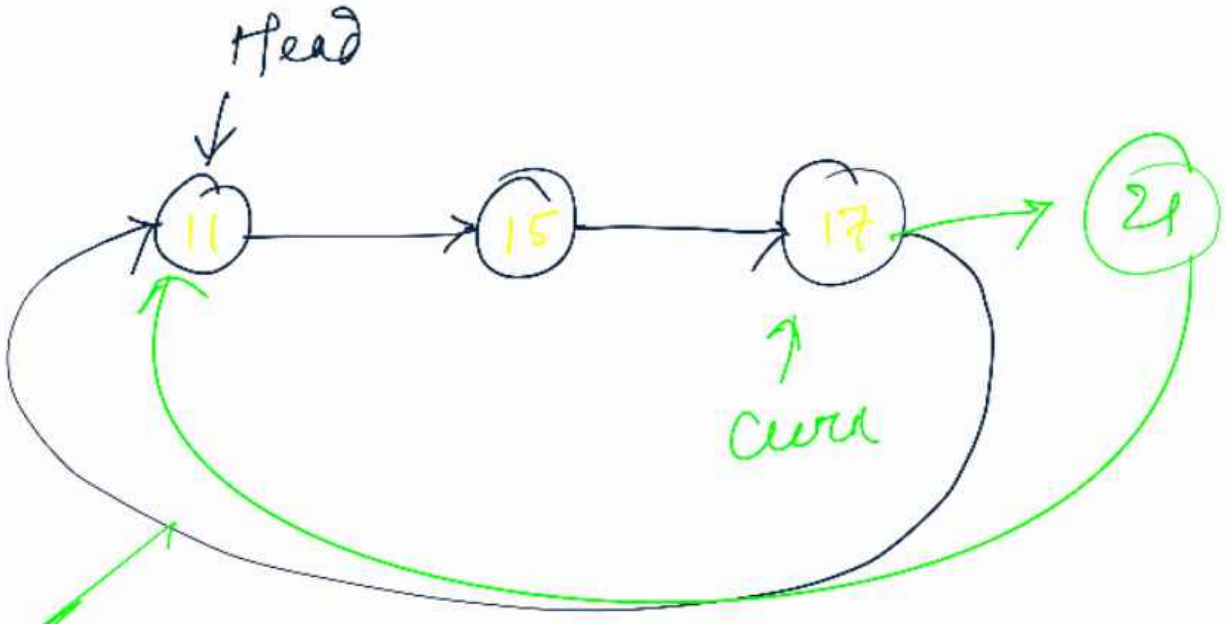
10

Case-2

35

Case-3

key: 90



Case-1

data = 9

curr.data >= data

(11 >= 9) ✓

nn.next = head;

return

Case-2

data = 16

(15 < 16) ✓

(17 < 16) ✗

Case-3

data = 21

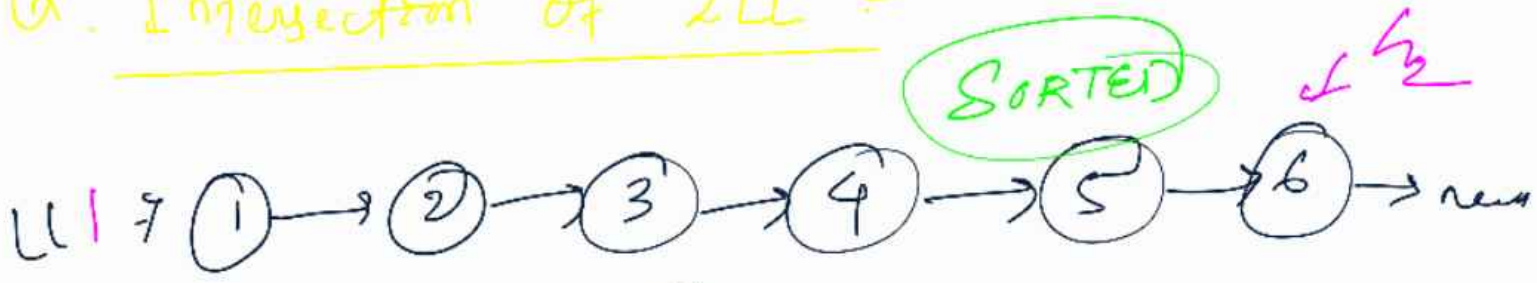
(15 < 21) ✓

(17 < 21) ✓

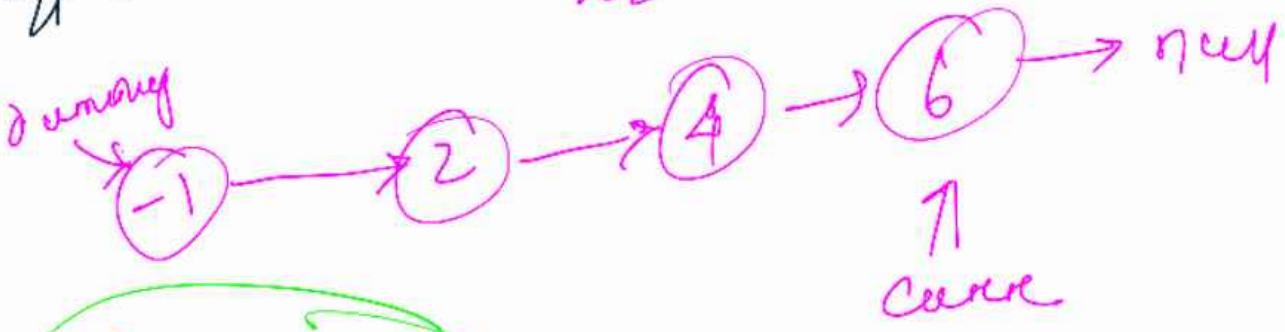
(11 < 21)

Q. Intersection of 2LL :-

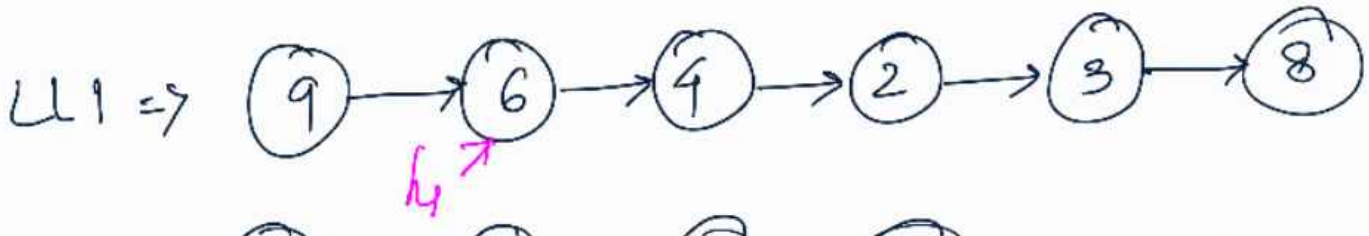
SORTED

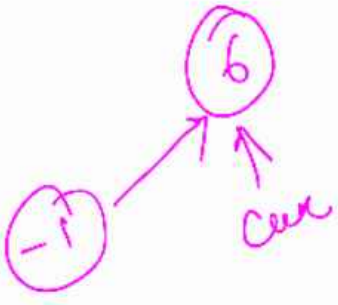


O/P :



UNSORTED

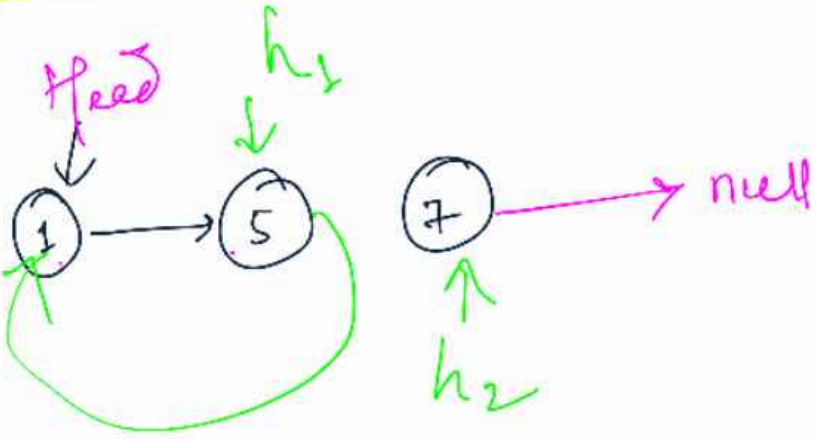


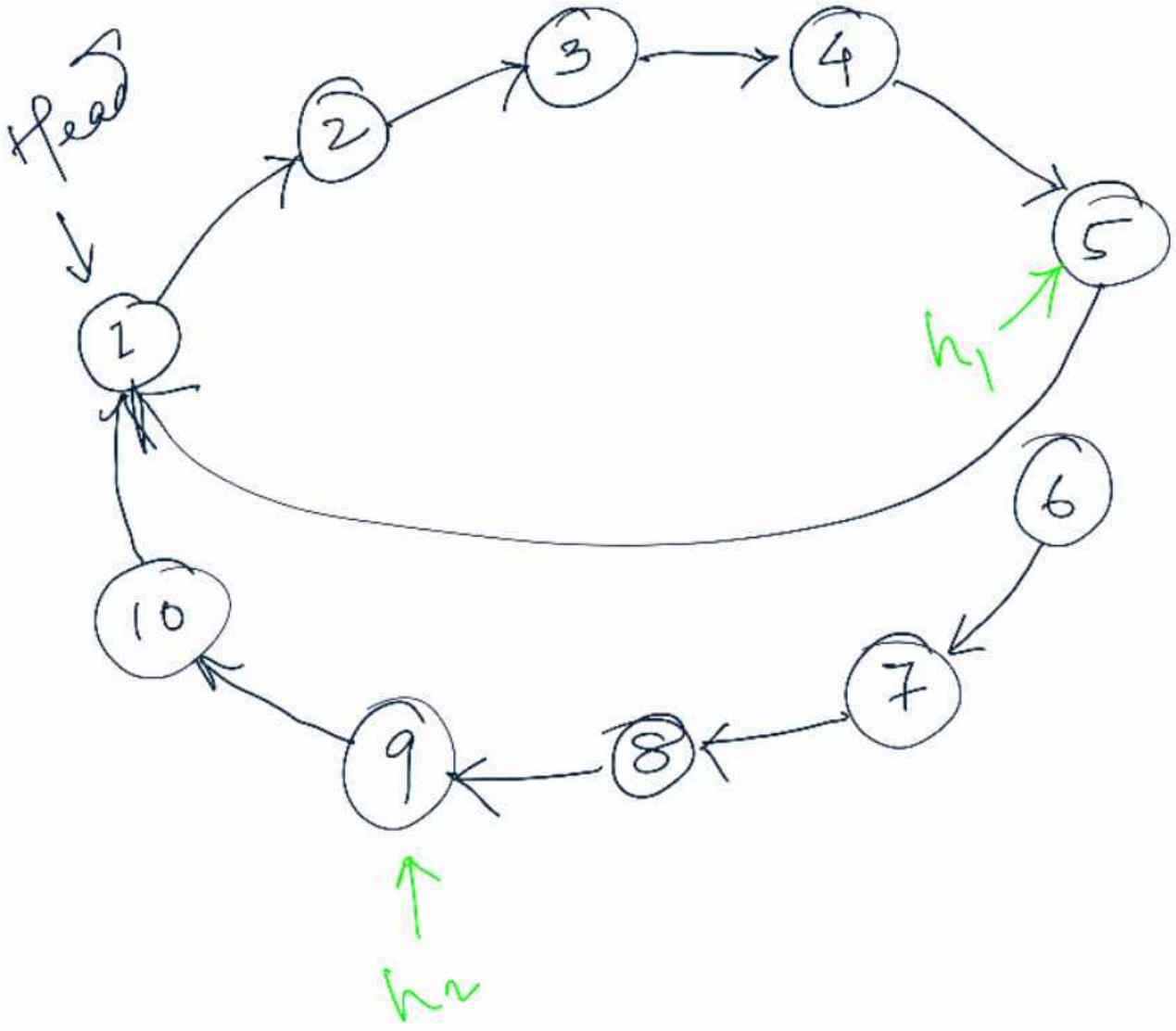


hm

{
1 : 1
2 : 1
8 : 1
6 : 0
}

Q. Split a Circular LL into two halves :-



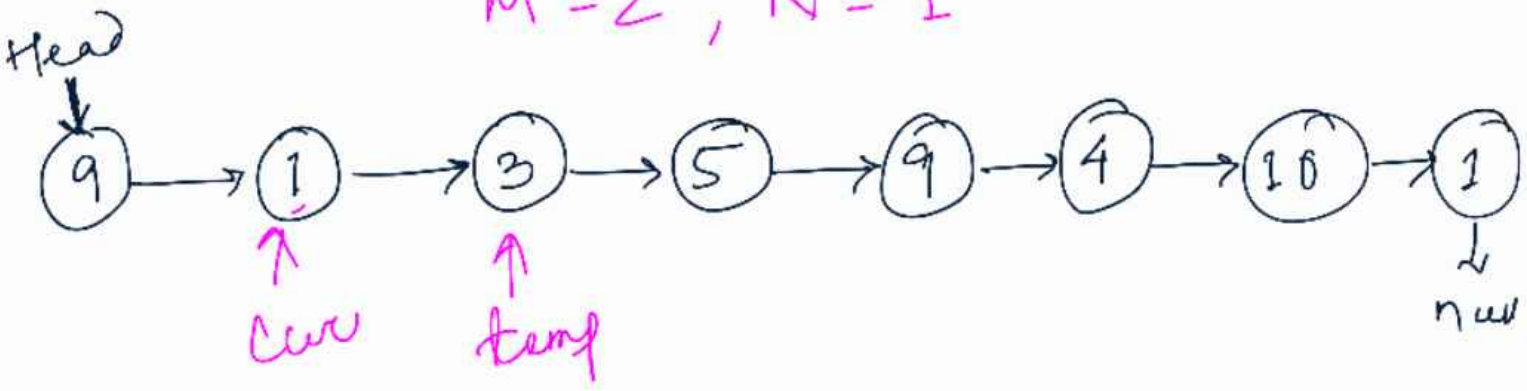


[per day → 20-40 ques
need to solve

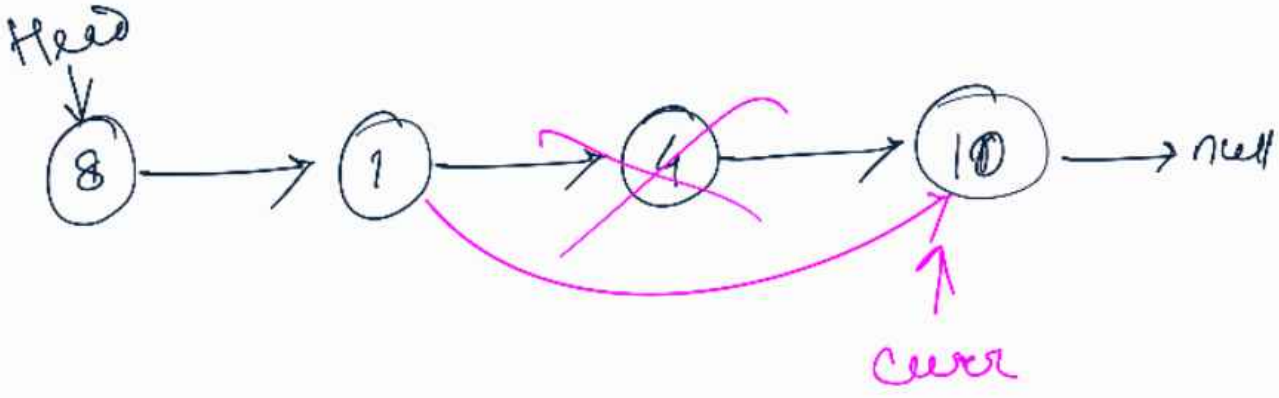
15 hrs
= 30 ques

Q. Delete N nodes after skipping M nodes.

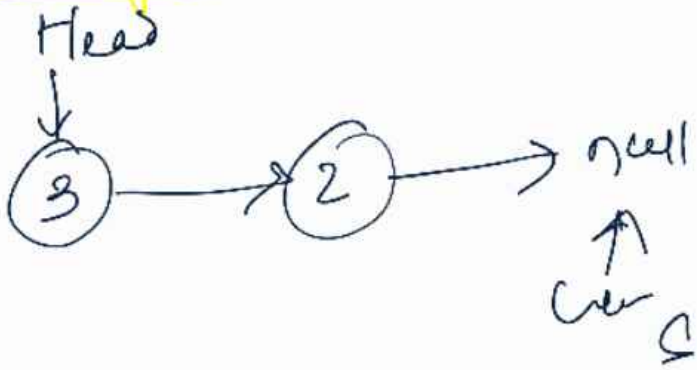
M = 2 , N = 1



cm = 2 cn = 1



Q. Multiply Two LL



sum = 0 3

Scurr = sum * 10 + curr.data

3 * 10 + 2

= 32

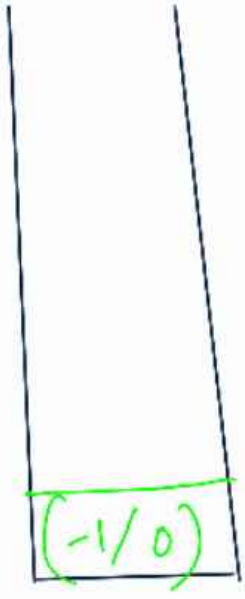
Q. Make array beautiful :-

arr [4 -2 2 1 -1 1]

[4 1] ⇐ arr

[2, 1, -4, 3 -5, 2, -6, 3] o/p

[2, 2]



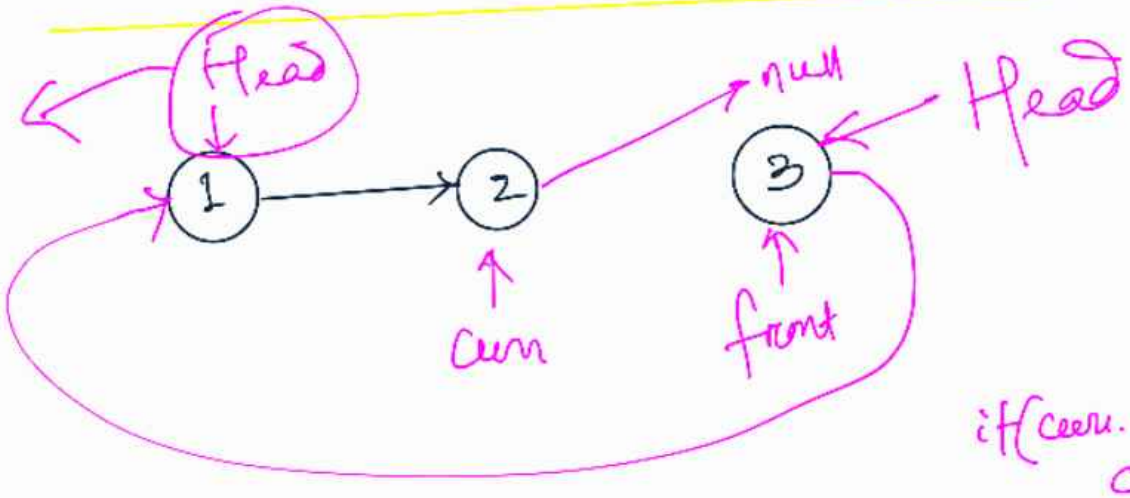
-7

7

[-3, -1, -19, 0, 6, -13, 12]

[(-1/0), 1]

Q. Move last to front on LL :-




```
if(curr.next.next != null)
    curr.next;
curr.next = null;
```

Q. Special Digit :-

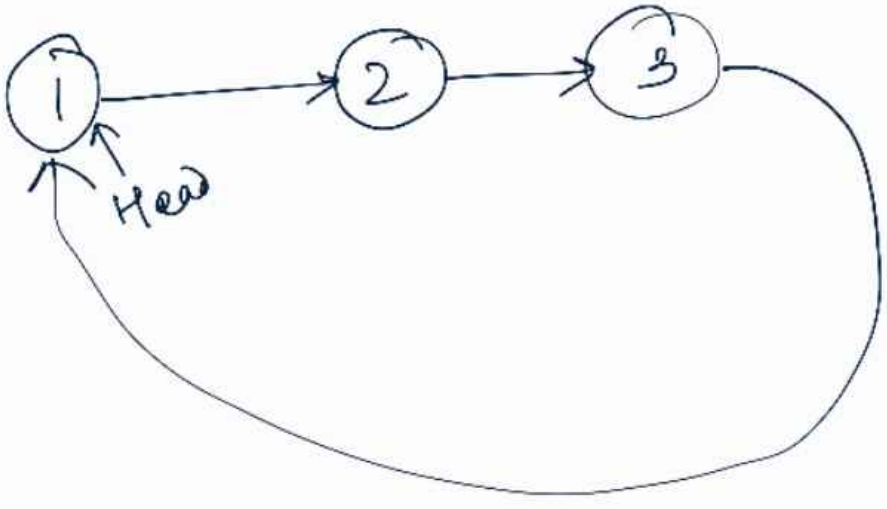
$N = 2$

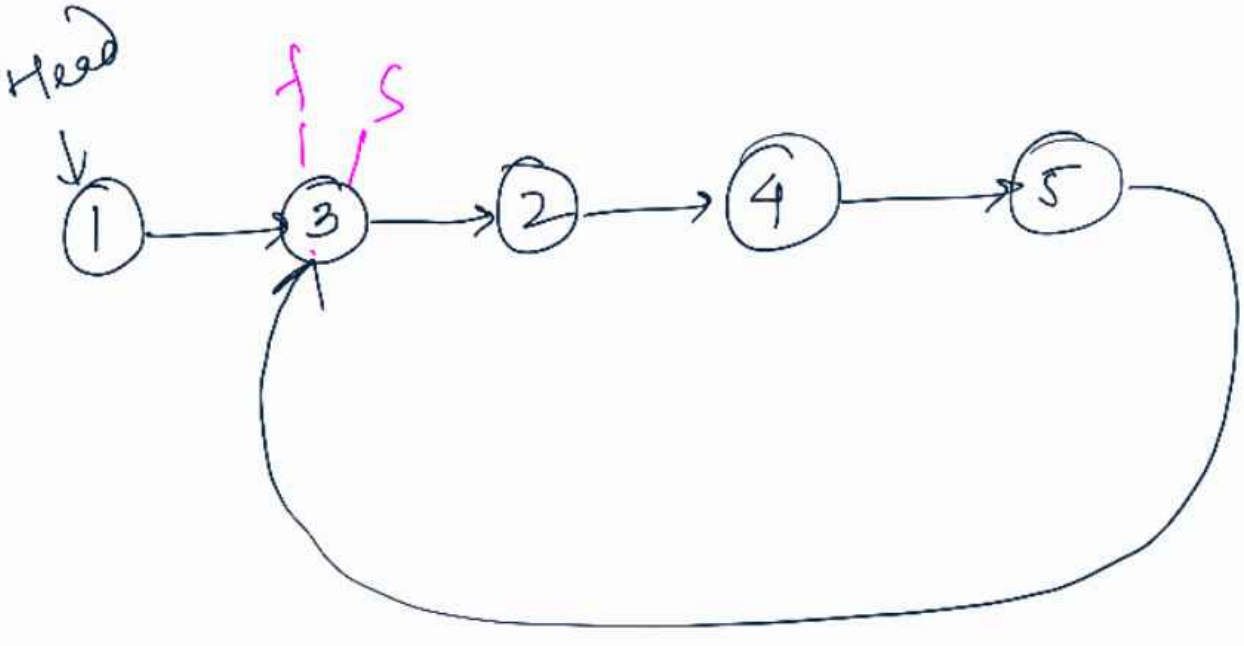
$A = 1$ $\quad C = 3$

$B = 2$ $\quad D = 5$

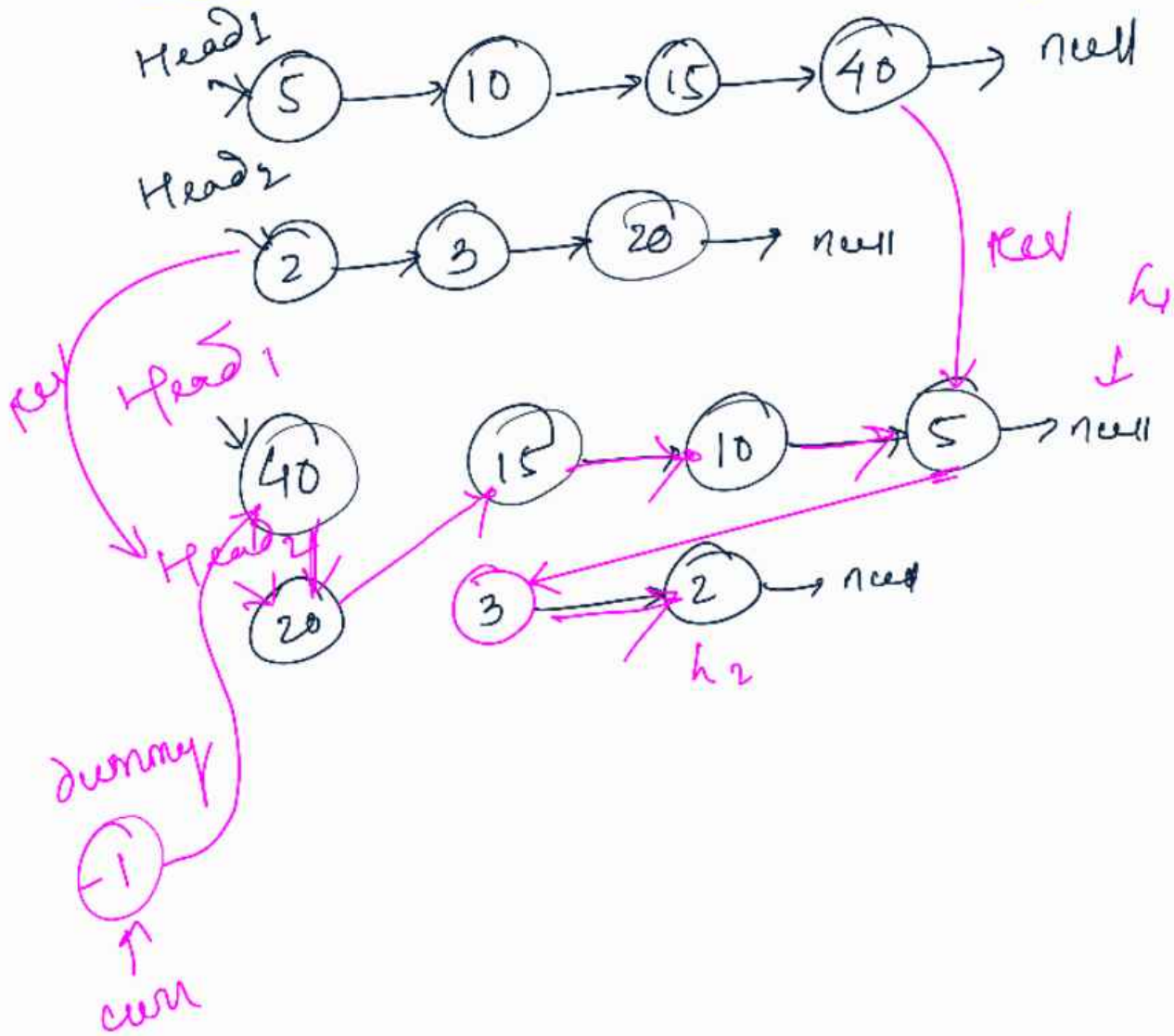
[11, 22, 12, 21]

Q. Find the first node of loop in LL :

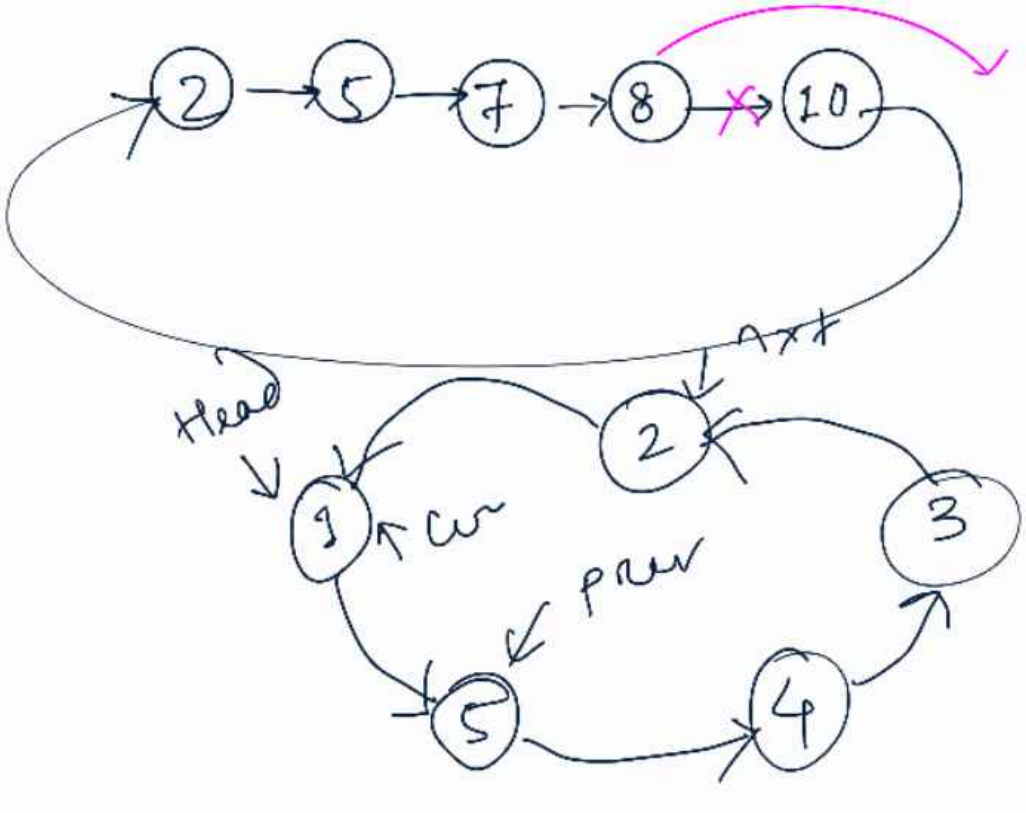




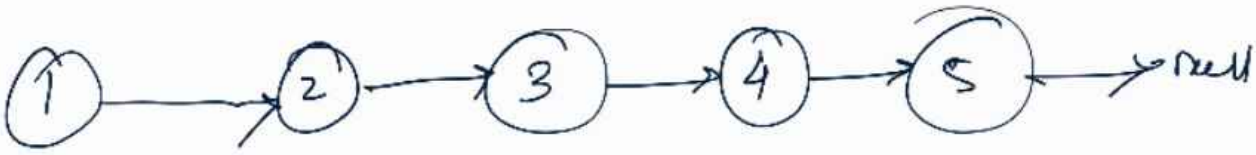
Q. Merge 2 Sorted LL in reversely.



Q. Delete and Revers CLL :—



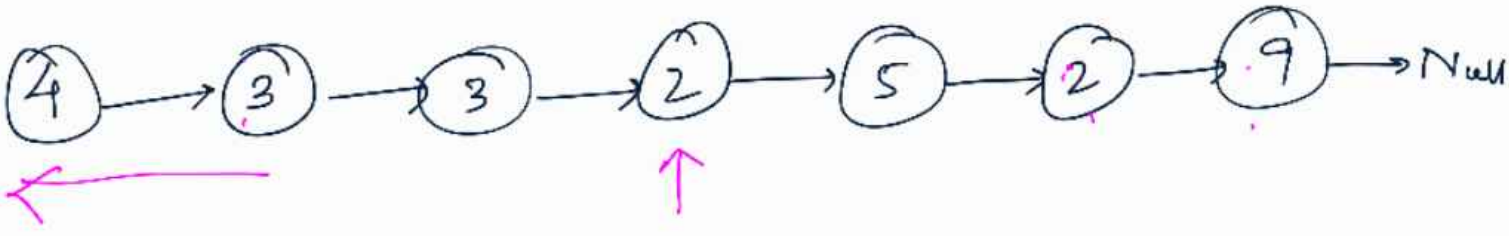
Q. Reverse Both Part :



Case-1

Case-2

Case-3

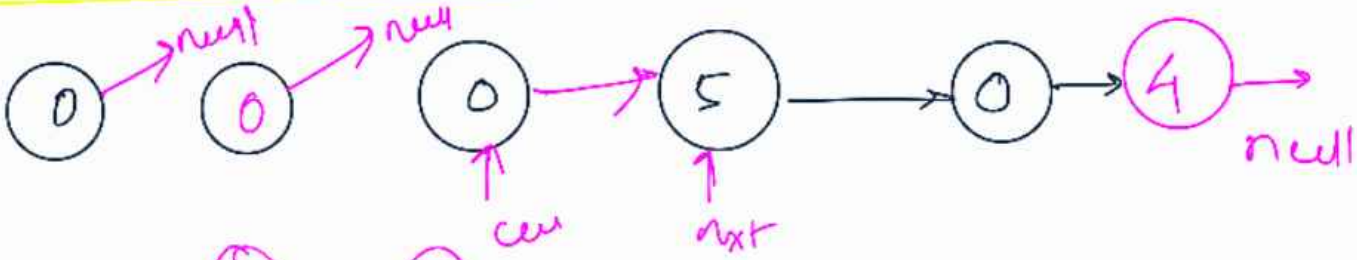


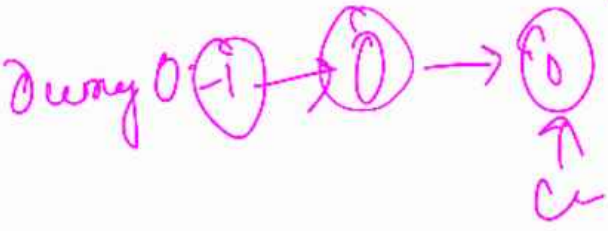
Q. Zig - Zag :-

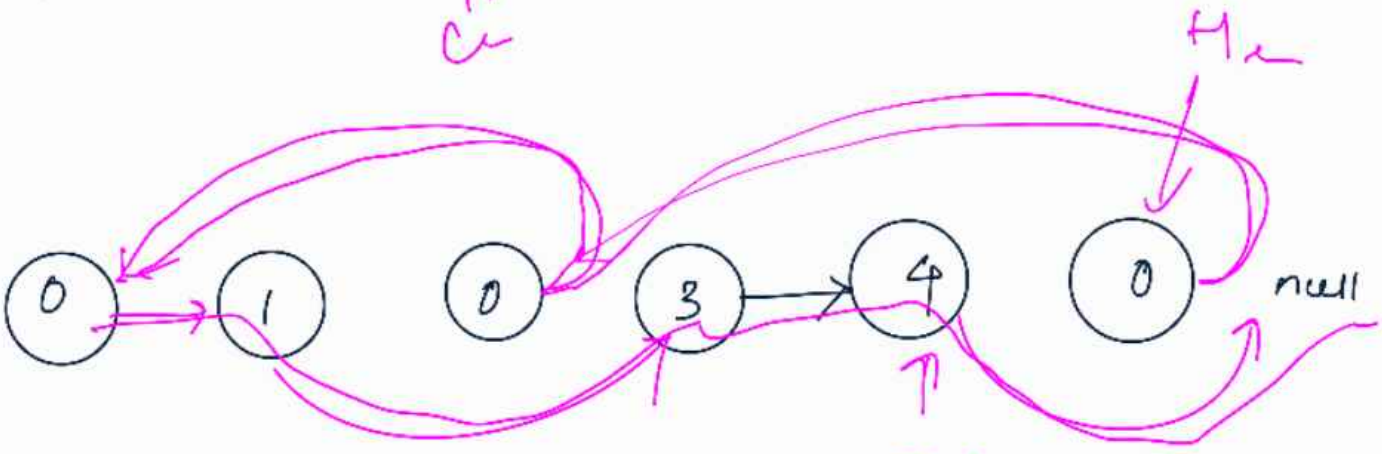
(pending)

temp

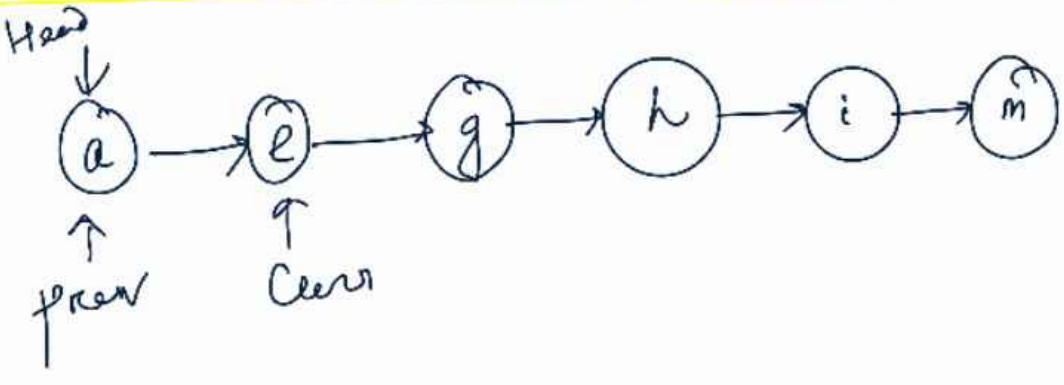
Q. Move Zeros to front :-





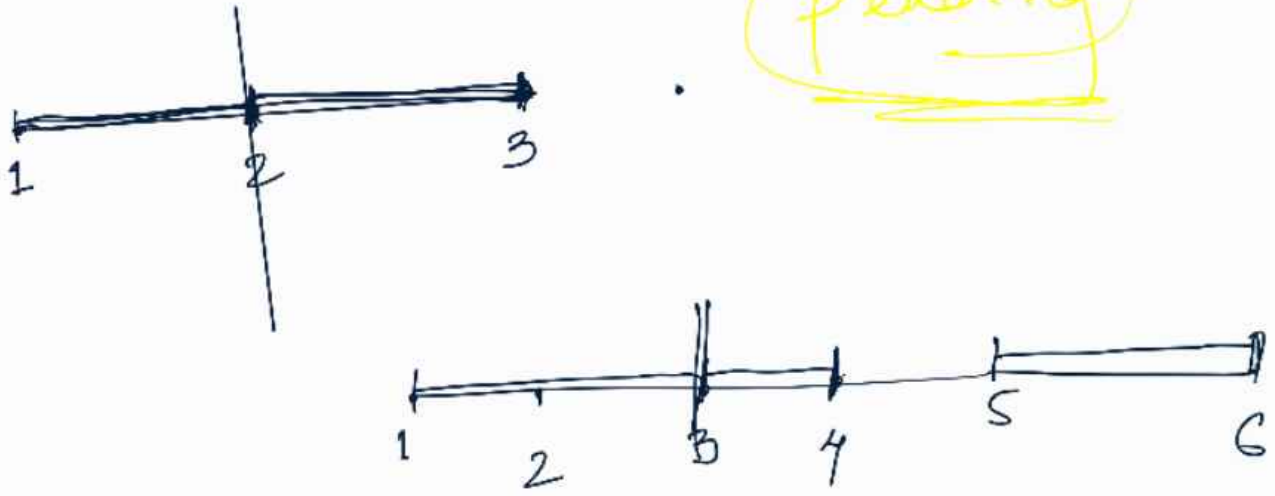


Q. Arrange Consonat and Vowels

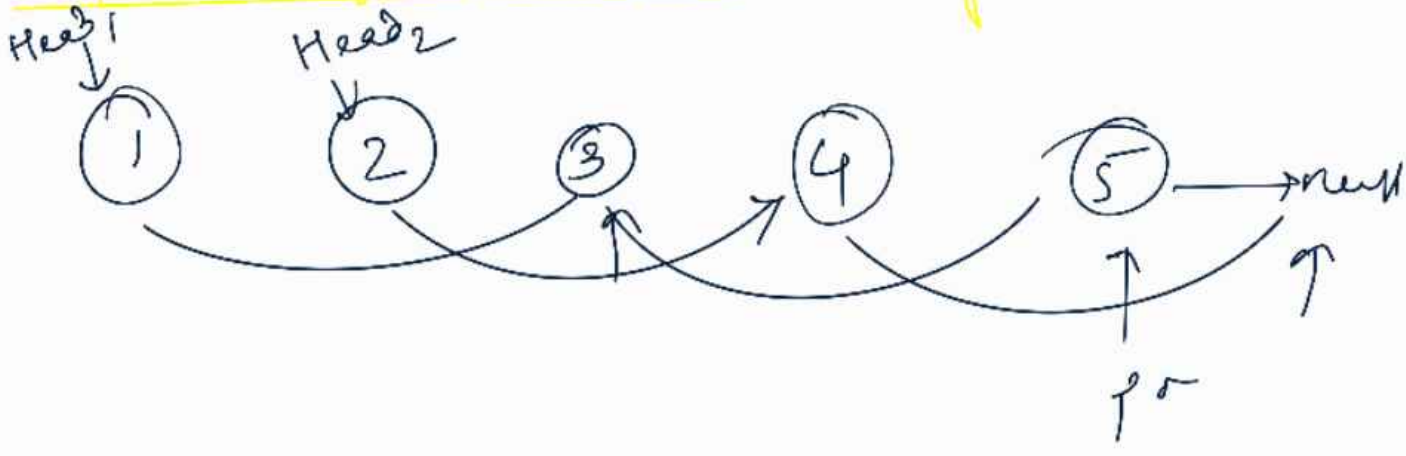


Q. Maximum Intersecting Lines :-

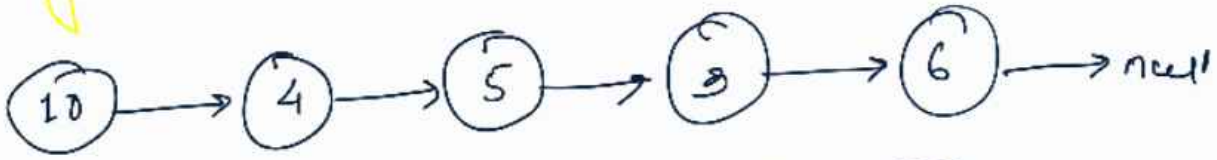
pending



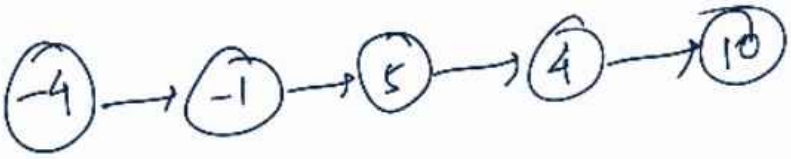
Q. Split Single LL alternatively :-

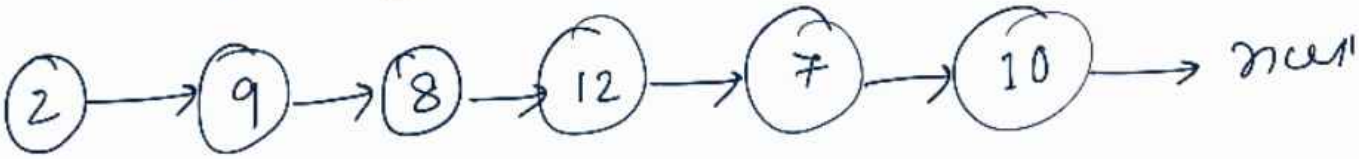


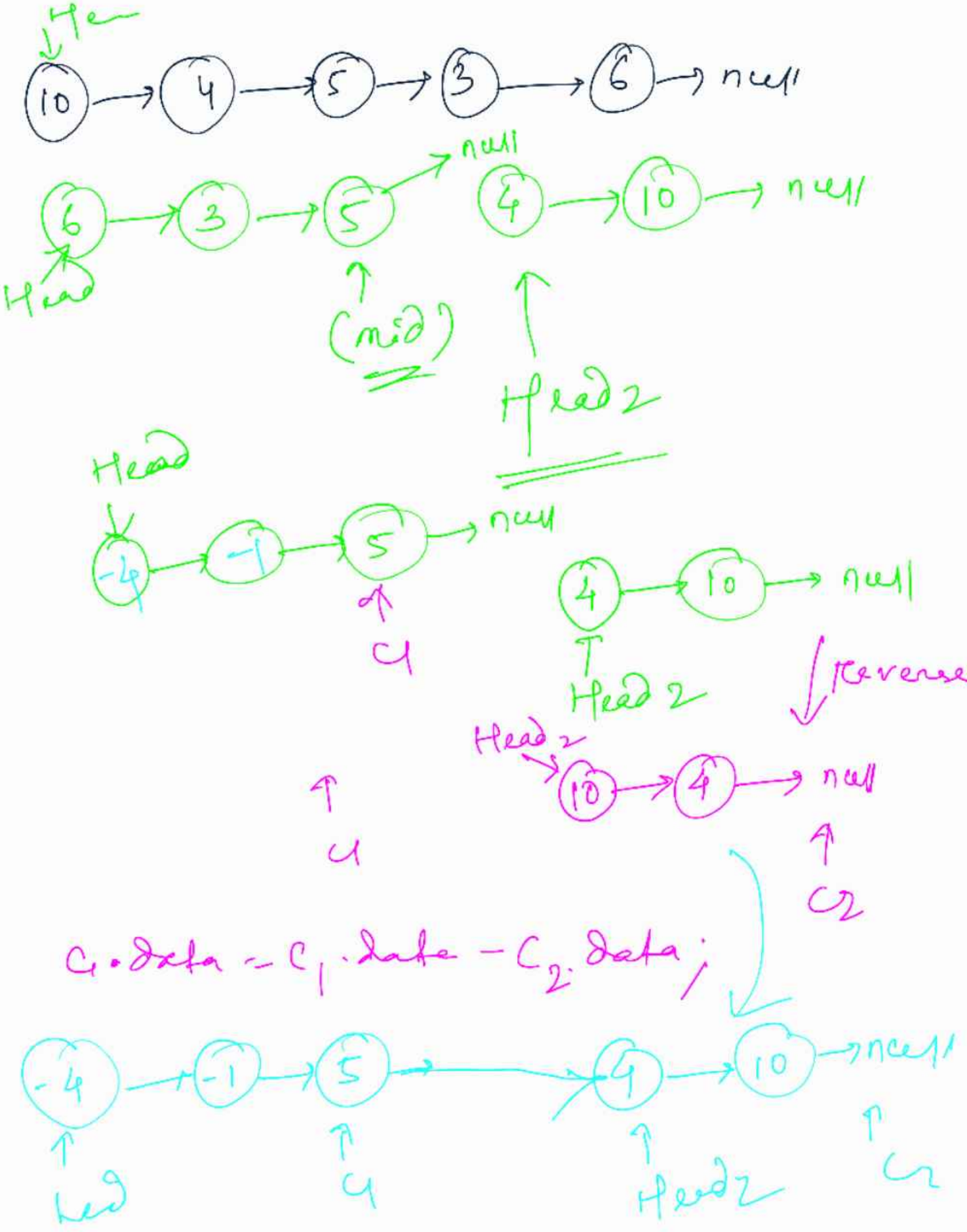
Q. Modify LL-1 :-

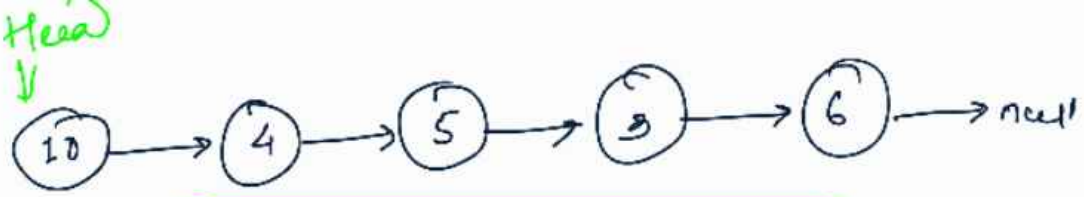


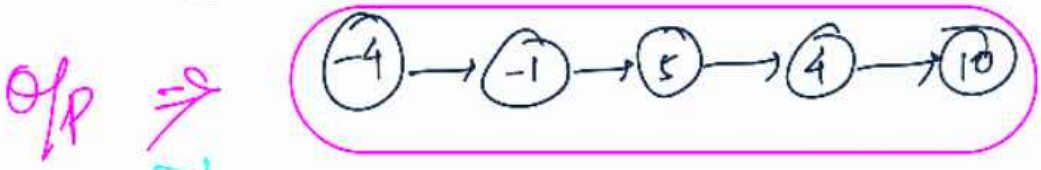
ans ⟹

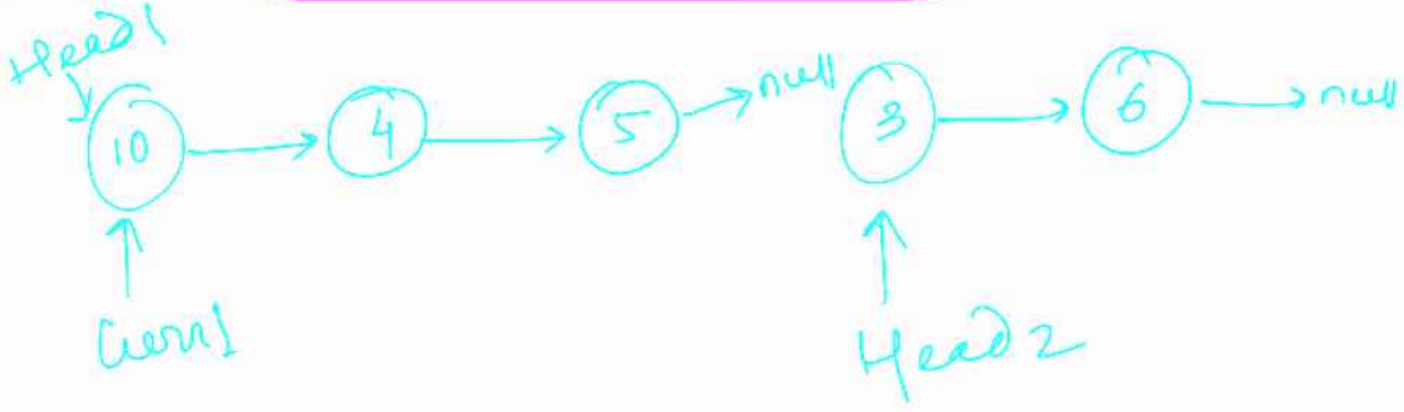


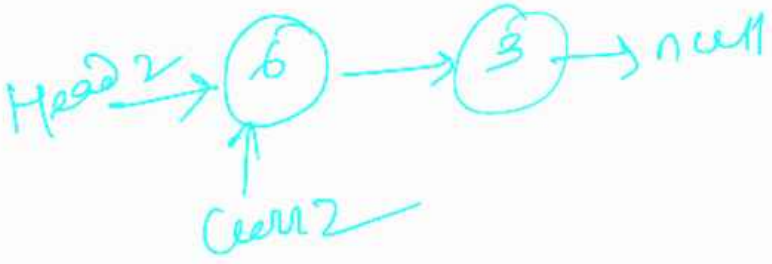



10
4
5
3
6
null
6
3
5
null
4
10
null
Head
(mid)
Head 2
Head
-4
-1
5
null
4
10
null
C1
Head 2
Reverse
Head 2
10
4
null
C1
C2
C1.data = C1.data - C2.data;
-4
-1
5
4
10
null
Head
C1
Head 2
C2



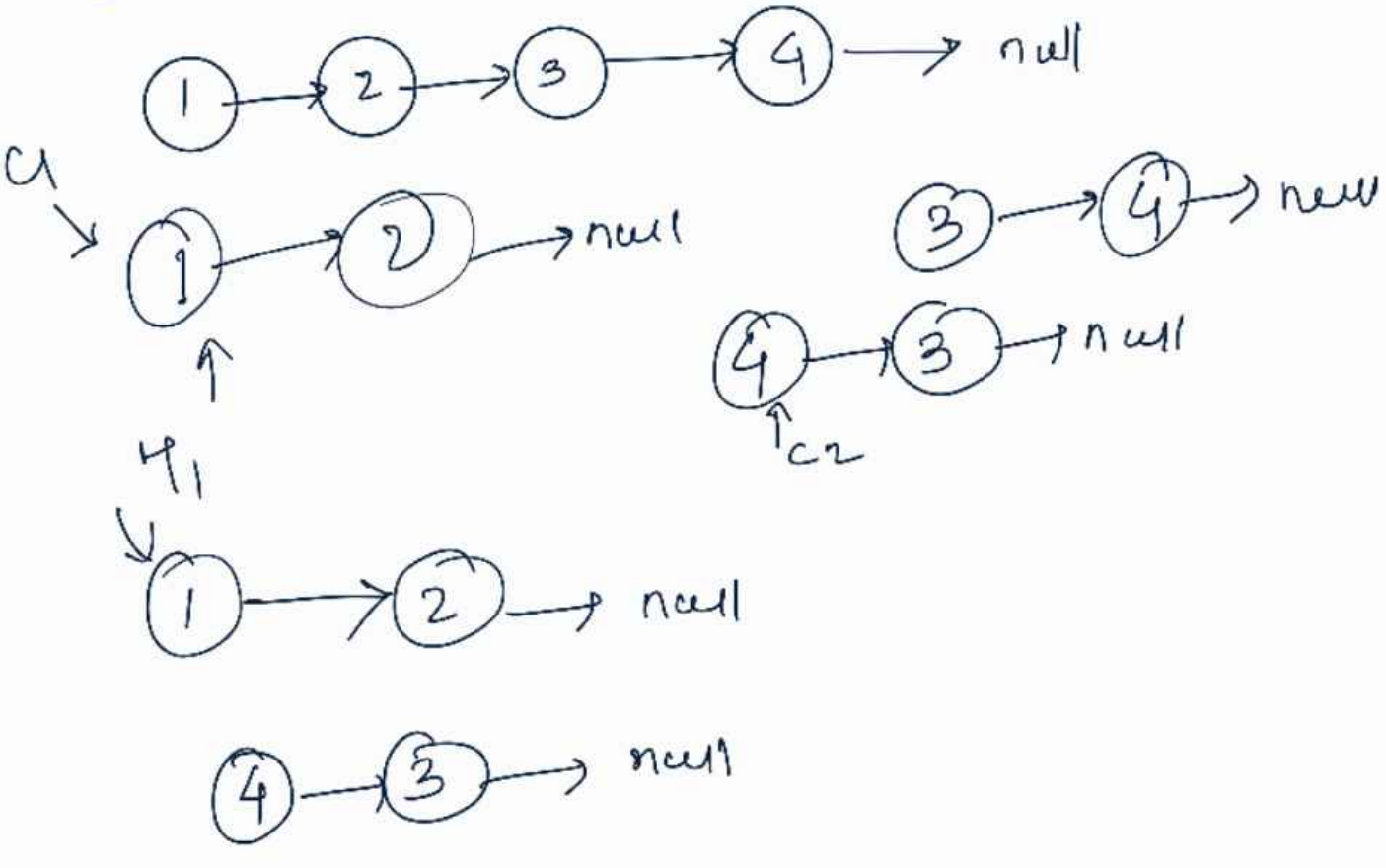


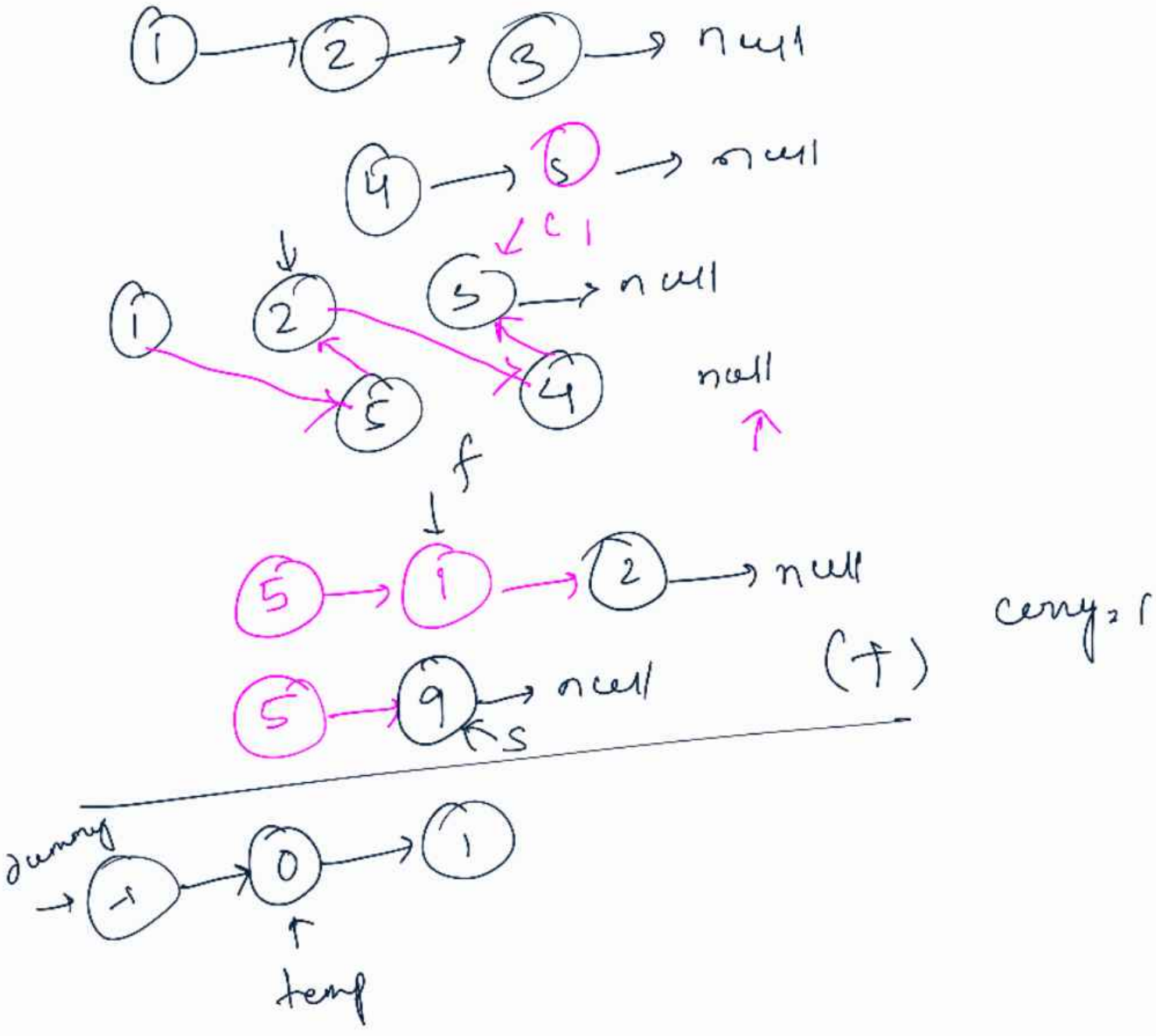




int temp = 6

Q. Rearrange LL




1
2
3
null
4
5
null
c 1
1
2
5
null
3
4
null
f
5
1
2
null
carry = 1
(+)
5
9
null
s
dummy
-1
0
1
temp