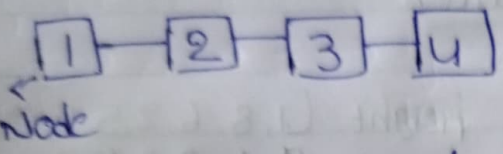
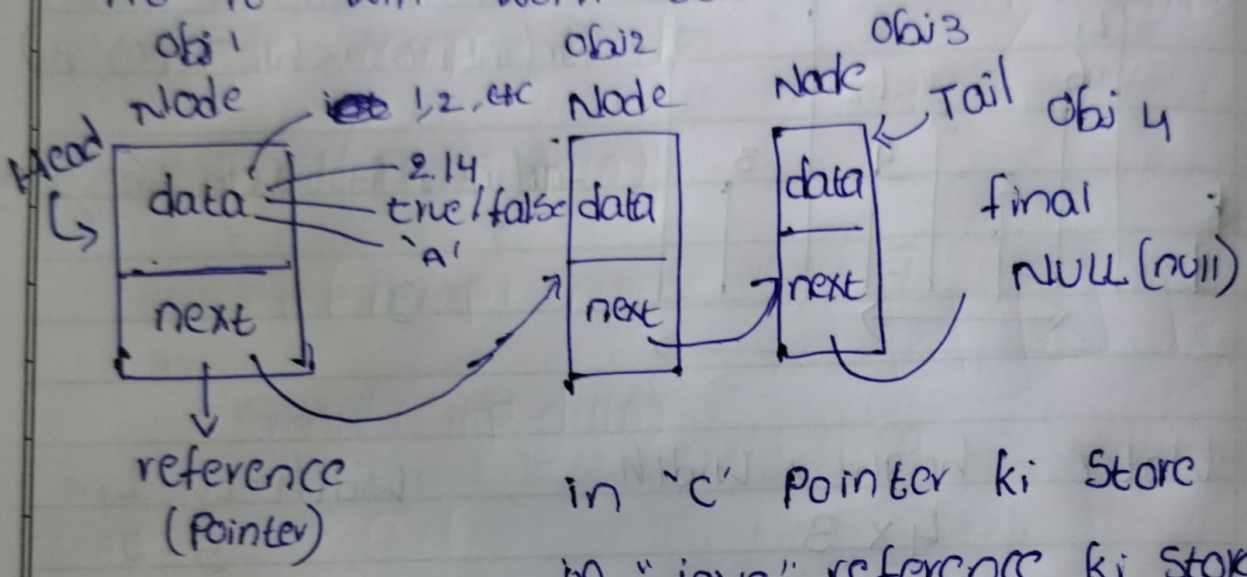
linked list → Single linked list



How it will work structure



in 'C' Pointer ki Store

in "java" reference ki store

reference (OOPS) → reference variables are variables that point to object.

↳ Node

Node → data, next } Blue Print

How to impliment

```
public static class Node {
    int data;
    int next;
}
```

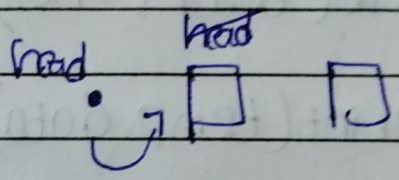
Date: .....

Expt. No.: .....

Page No.: .....

```
// constructor
public Node(int node) {
  this.data = data;  initialize
  this.next = null;
   }
   }
```

Add in linked List



Add First - O(1)

Add last

step to follow

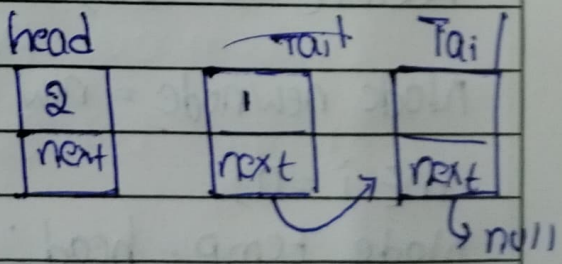
① To create a new node

② new node's next = head

③ head = new node

Add last

step to follow



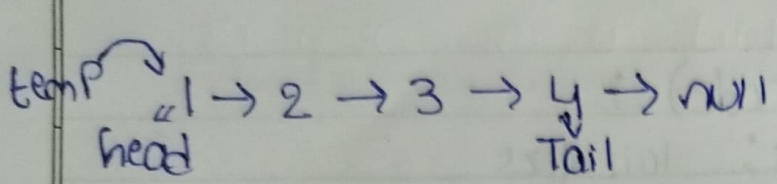
① create a node (new node)

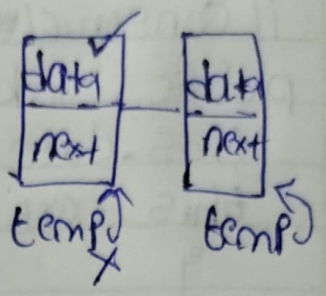
② tail.next = new node

③ tail = new node

Teacher's Signature _____

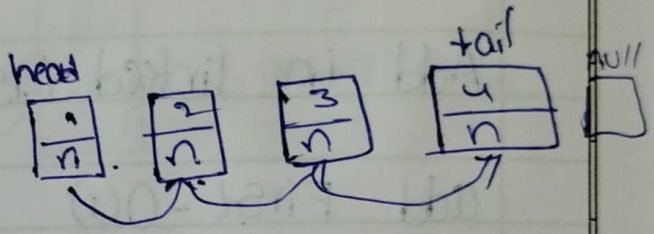
Print a linked List






```
Node temp = head;

print(temp.data)

while(temp != null){

  print(temp.data)

temp = temp.next;
```

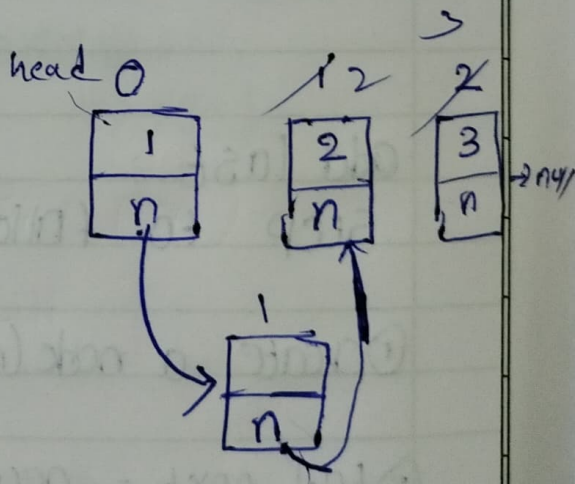
y



add Middle

```
AddM(int idx, int data)
Node newnode = new Node(data)
int i = 0;
Node temp = head;
while (i < idx - 1)
temp = temp.next;
i++
```

y



Date: .....

Expt. No.: .....

Page No.: .....

temp(prev)

newnode.next = temp.next;
temp.next = newnode;

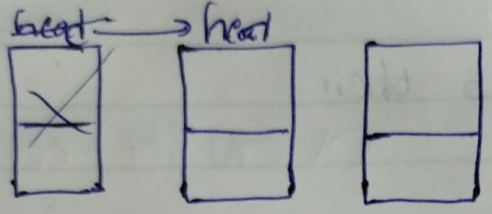
One special case is their

```
if(idx == 0) {
    AddF(data);
}
```

Teacher's Signature _____

Remove in a LinkedList

Remove First



head = head.next

Logic

```
RemoveF()
    int val = head.data;
    head = head.next
    return val;
```

→ sizec--; head = newnode;

Special Case

```
if (Siec == 0) {
    System.out.println("Empty List");
    return Integer.MIN-VALUE;
}
else if (size == 1){
    int val = head.data;
    head = tail = null;
    return val;
}
```

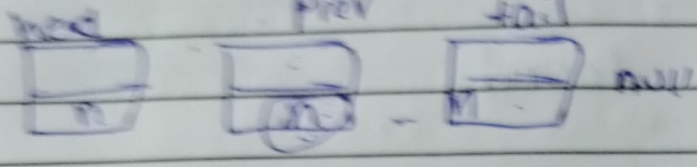
Size = 0;

Date: .....

Page No.: .....

Roll No.: .....

Remove last



Steps

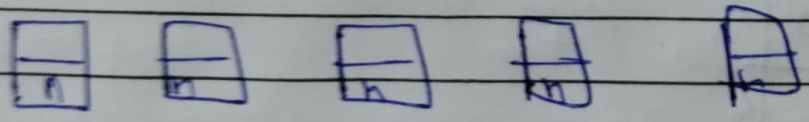
① prev.next = null

② tail = prev

```
// Prev → size-2
Node prev = head;
for (int i=0; i<size-2; i++){
    prev = prev.next
    }

int val = prev.next.data; // tail data
prev.next = null;
tail = prev;
size--;
return val;
```

Search (Iterative)

Search for a key in a linked list. Return the position where it is found. If not found, return -1



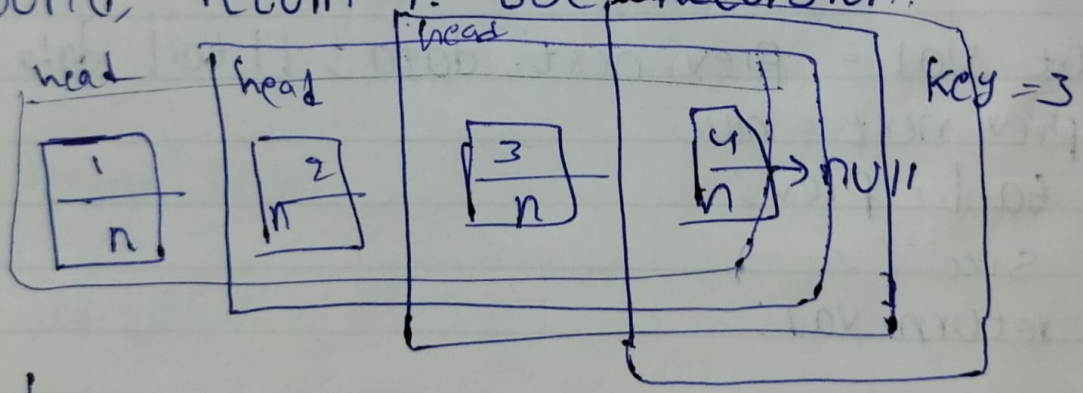

```
while(temp!=null){
    if (temp.data==key){
        return i;
    }
```

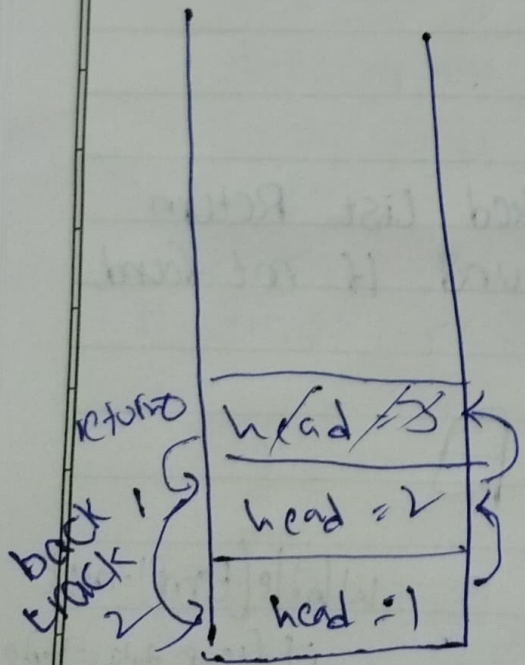
Teacher's Signature _____

```
temP = temP.next
 i++;
 y
retur -1 //not found
```

Search (Recursive)

Search for a key in a linked list. Return the position where it is found. If not found, return -1. use Recursion.





Date: .....

Expt. No.: ..... Page No.: .....

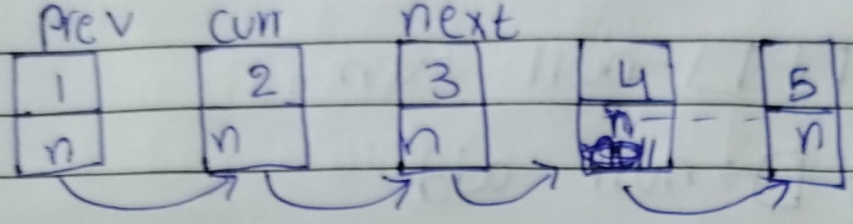
Reverse a Linked List

$1 \to 2 \to 3 \to 4 \to 5 \to null$

Iterative Approach

$null \leftarrow 1 \leftarrow 2 \leftarrow 3 \leftarrow 4 \leftarrow 5$

head



while(curr != null) ?

① next = curr.next

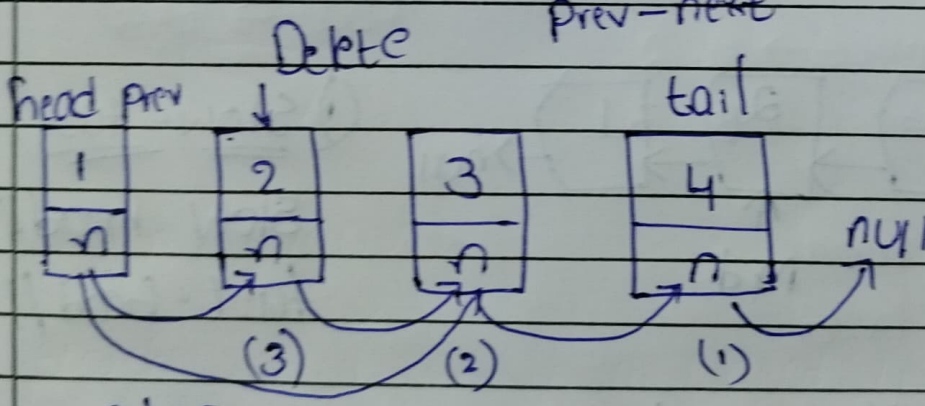
② curr.next = prev

③ prev = curr

④ curr = next

y

Find & Remove Nth node from End



N = 3

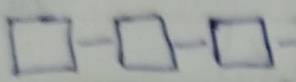
$(4 - 3 + 1) = 1 + 1 = 2$

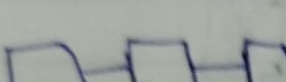
$(Size - n + 1)^{th}$ Start

Teacher's Signature _____

Check if LL is a Palindrome

racecar → racecar → it is a Palindrome

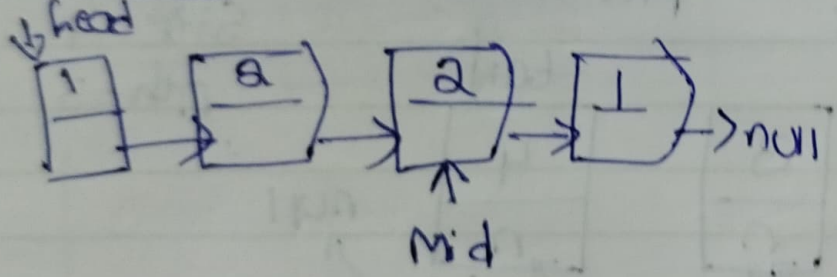




1) find the Mid Node
2) 2nd half Reverse
3) check if (1st half == 2nd half)
(left) (right)

Note
fast != null
fast.next != null

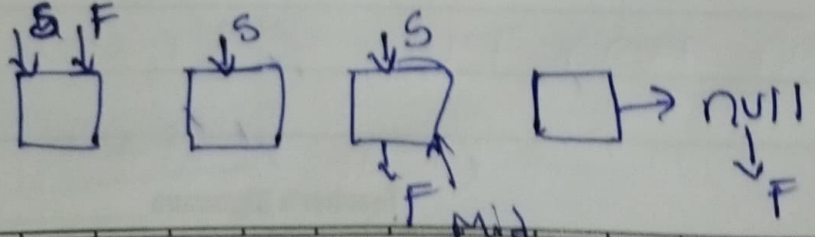
Even case



① slow - fast
slow = head // +1
fast = head // +2

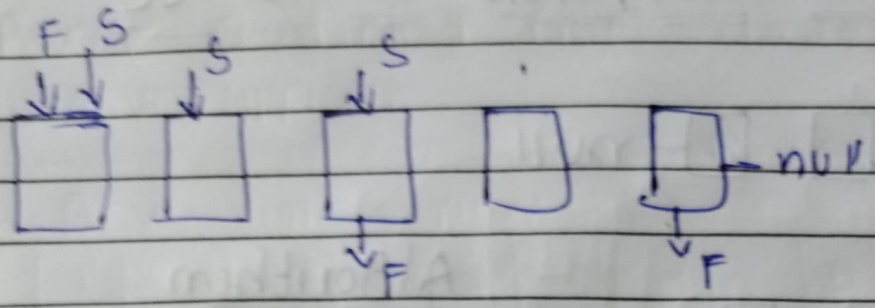
① find the Mid
slow = head ; // +1
fast = head ; // +2

(fast != null)



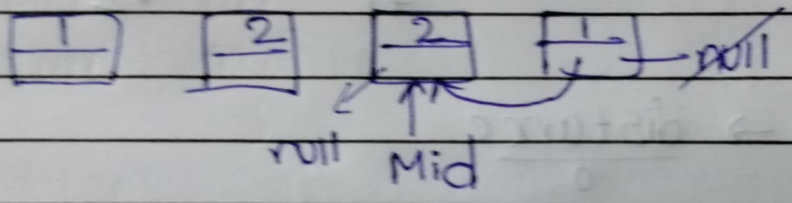
Date: .....

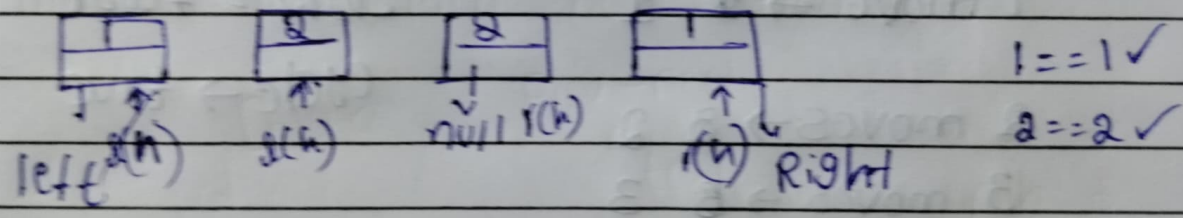
Expt. No.: ..... Page No.: .....



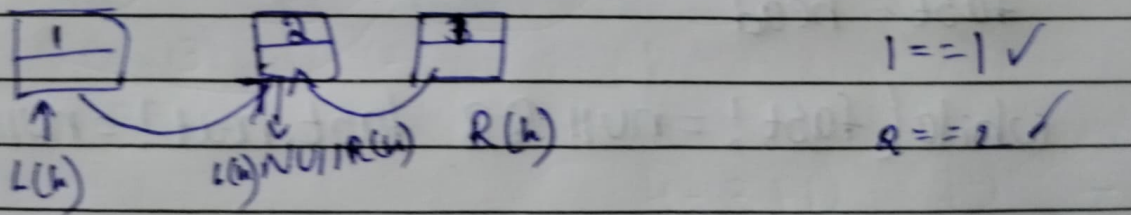
fast.next.!=null

2) 2nd half Reverse



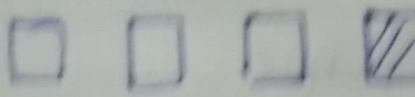


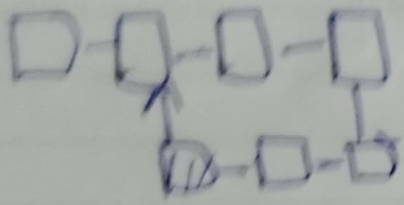
odd



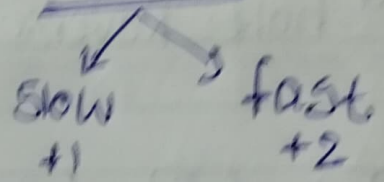
Teacher's Signature _____

Detect a Loop/cycle in a LL





Algorithem
Floyd's Cycle finding Algorithm



0 moves → distance 0

1 move → 1

2 moves → 2

3 moves → 3

Slow = fast
cycle → true

Slow = head
fast = head

while(fast != null && fast.next != null){

Slow → Slow.next // +1

fast → fast.next.next // +2

if(Slow == fast) {

return true
}
return false;

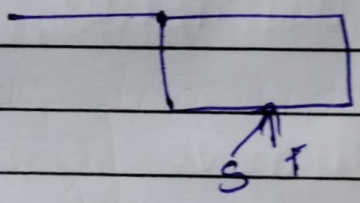
Remove a Loop/cycle in a LL

1) find last node

2) last node. next = null

Algorithem / approach

① detect cycle



a) slow = head

b) ~~slow → +1~~ prev = null
~~fast → +1~~

c) while (slow == fast)?

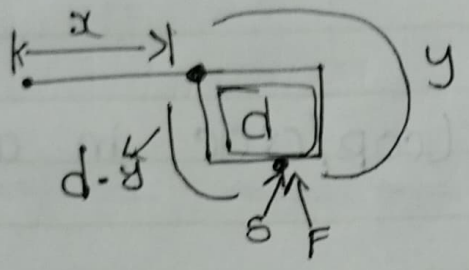
slow → +1 prev = fast
fast → +1

}

prev. next = null

Remove a loop/cycle in a LL

Explaination



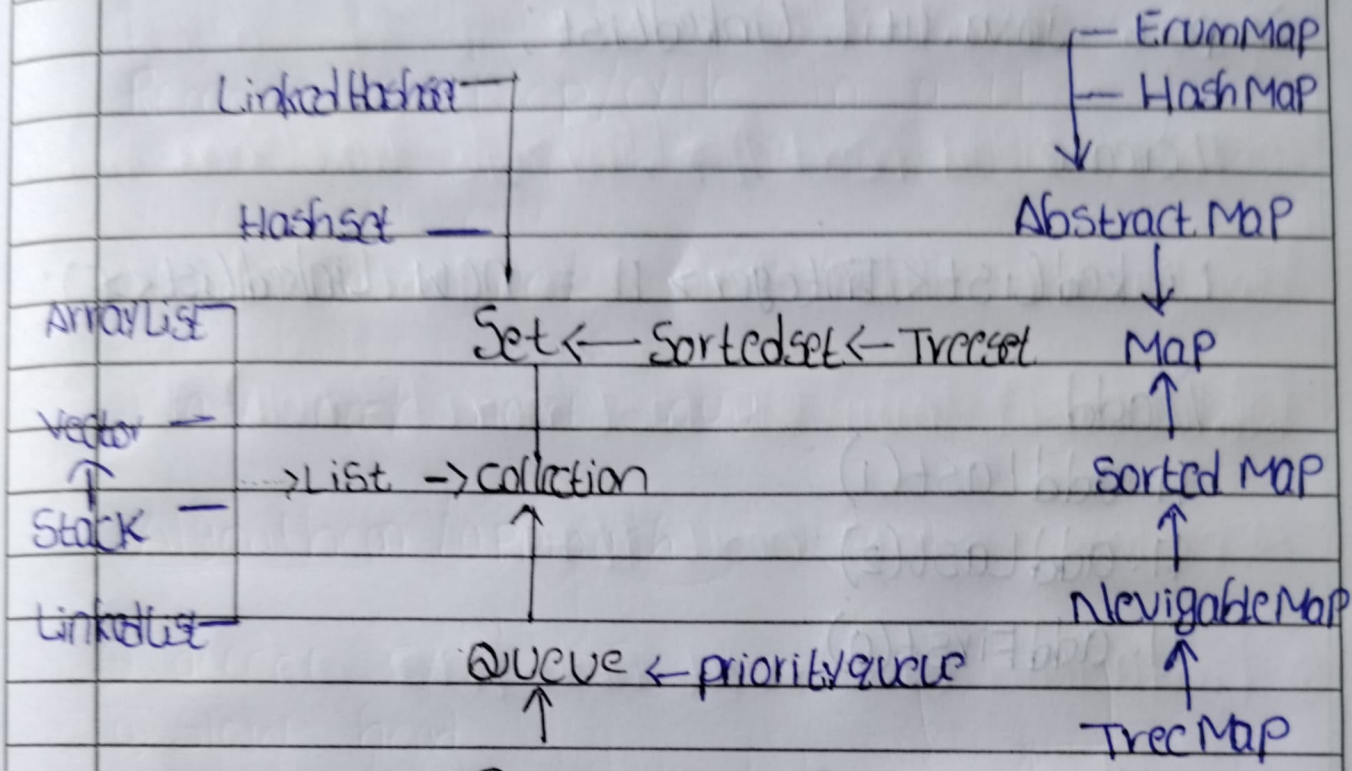
1)

Date: .....

Expt. No.: .....

Page No.: .....

Java collections Framework



JCF → ArrayList
LinkedList
Stack
Queue
HashSet
HashMap
→ Mostly Used

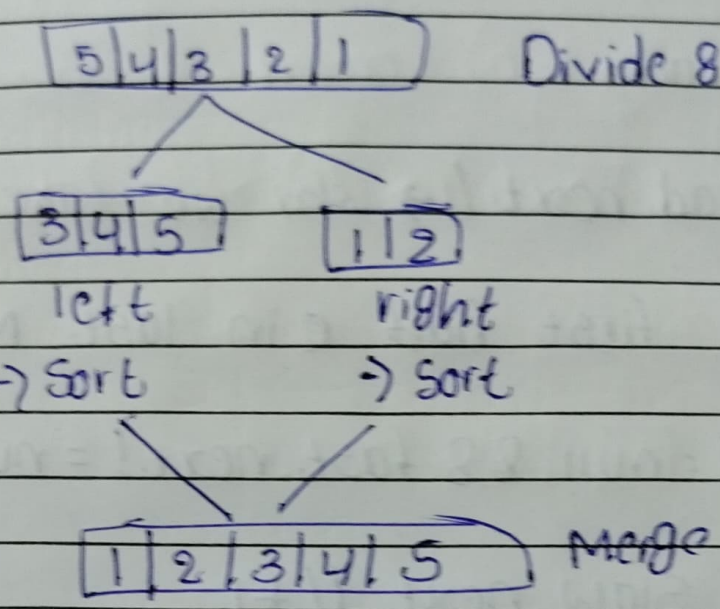
Teacher's Signature _____

LL in Java Collections Framework

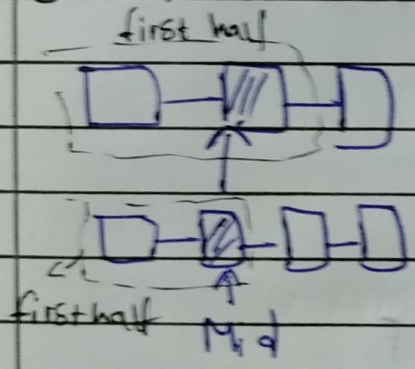
```
import java.util.LinkedList;

  // create
  LinkedList<Integer> ll = new LinkedList<>();

  // add
  ll.addLast(1);
  ll.addLast(2);
  ll.addFirst(0);

    // remove

    ll.removeLast();
    ll.removeFirst();
```

Expt. No.: ..... Page No.: .....

Mergesort on a Linkedlist



① LL → Mid



② i) left half ← MS

ii) right half ← MS

Mid next = null

③ Merge

Teacher's Signature _____

logic

① Mid

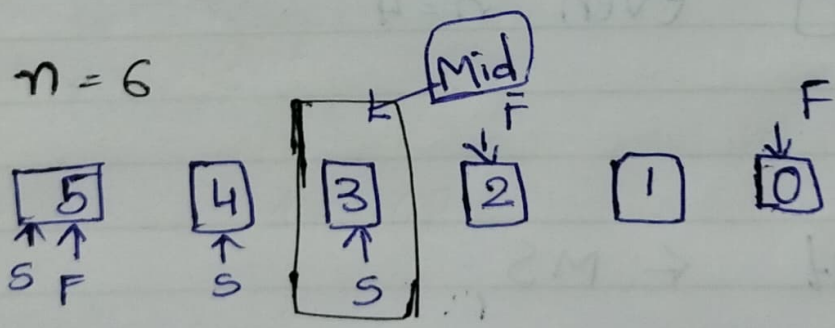
slow = head

fast = head.next // → why we take next means we want first half @ in last Mid.

```
While(fast != null && fast.next != null) {
    slow = slow.next //+1
    fast = fast.next.next //+2
}
```

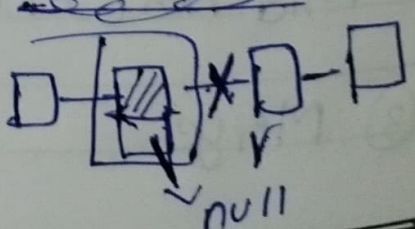
slow → Mid Node

Ex: n = 6



②

```
rightHead = Mid.next;
Mid.next = null;
```



Date: _____

Book No.: _____ Page No.: _____

MS (head1) // left half

MS (right head) // right half

③ head1 = next head2 = next

Node MergedLL = new Node(-1);

Node temp = MergedLL

while (h1 != null && h2 != null) {

h1 <-> h2 // compare

}

return MergedLL.next

Ex:

5 → 4 → 3 → 2 → 1

3 → 4 → 5 → null 1 → 2

temp

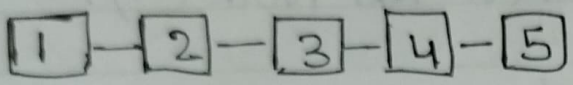
temp.next

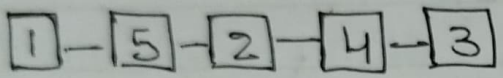
-1 → 1 → 2 → 3 → 4 → 5

Teacher's Signature _____

Zig-zag Linked List

For a linked list of the form L(1) -> L(2) -> L(3) -> L(4) .. L(n-1) -> L(n) convert it into a zig-zag form i.e. : L(1) -> L(n) -> L(2) -> L(n-1) -> L(3) -> L(n-2) ..

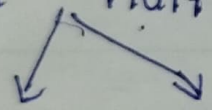




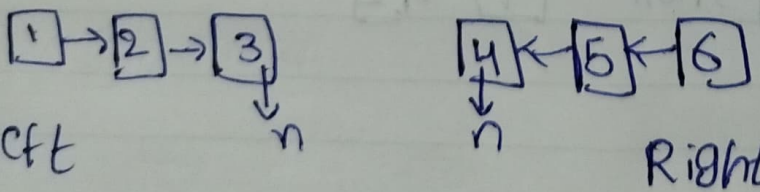
APproach

① find Mid (Mid = 1st half last node)

② 2nd half reverse

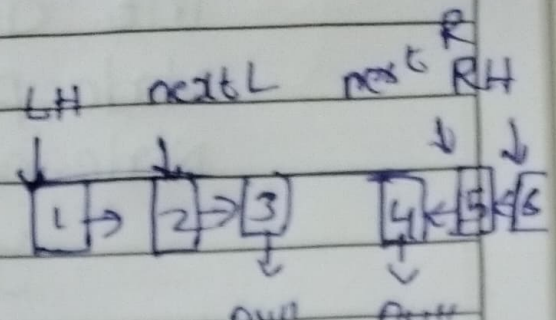


③ alternate Merging



Lefthalf.next = Right

Node LH = 1st half head
Node RH = 2nd half head



temp variable
Node next L; Node next R;

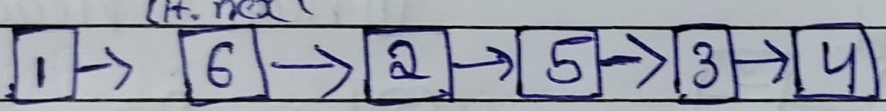
While(LH != null && RH != null) {

nextL = LH.next
~~nextR~~ LH.next = RH
nextR = RH.next
RH.next = nextL

}

zig-zag

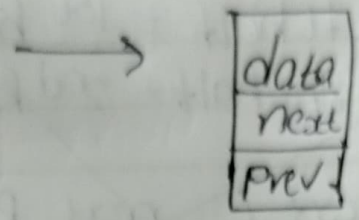
RH = nextR
LH = nextL
update

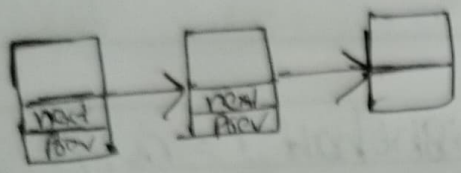


Teacher's Signature _____

Double linked list

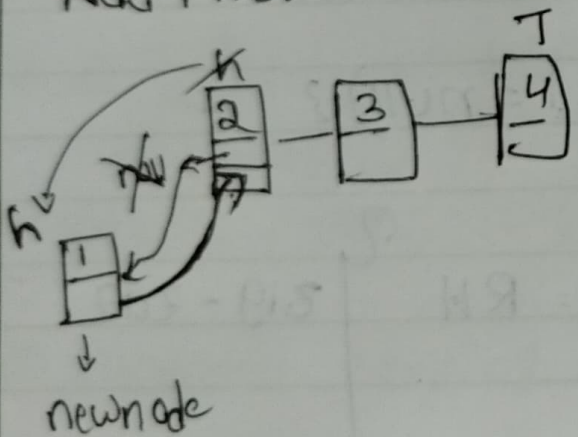
Approach






```
Public static class Node {

    int data;
    Node prev;
    Node next;
        }
```

Add First



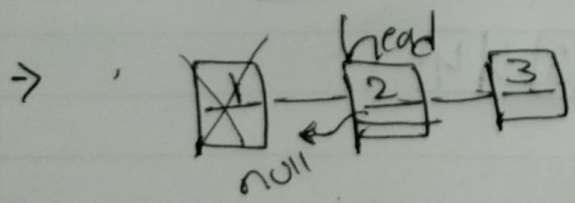
① create new node

newnode.next = head

head.prev = newnode

head = newnode

Remove First



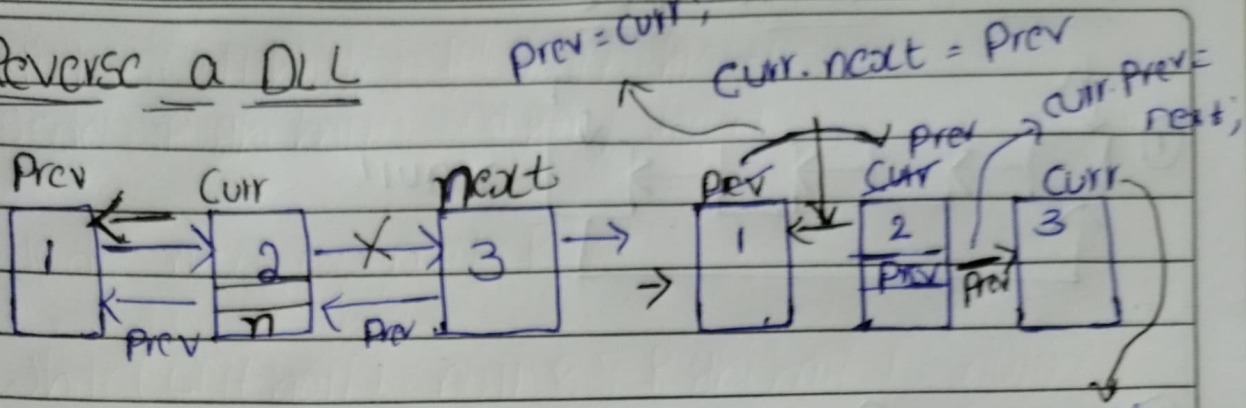
① head = head.next

② head.prev = null

Date: .....

Expt. No.: ..... Page No.: .....

Reverse a DLL



3 Variables four Steps

```
Node curr = head;
Node prev = NULL;
Node next;

// four steps

While(curr != null){

 next = curr.next;
 curr.next = prev;
 curr.prev = next;
 prev = curr;
 curr = next;

}
```

Note:- extra line

```
curr.prev = next
```

Teacher's Signature _____

Circular linked list