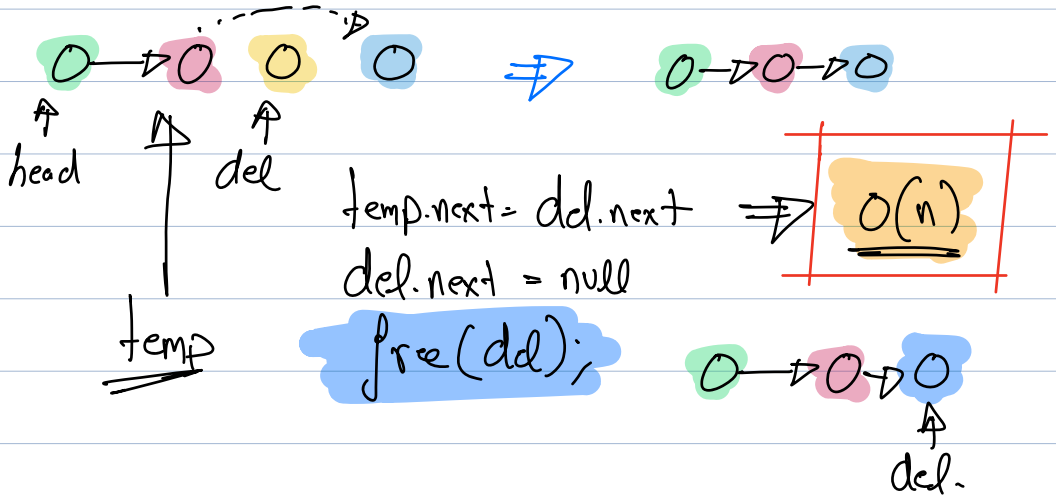
```
Node merge (Node head1, Node head2) {

    Node dummy = new Node(-1);
    Node temp = dummy;
    int i=0;
    while (head2 != null) {
        if (i%2==0) {
            temp.next = head1;
            head1 = head1.next;
            temp = temp.next;
        } else {
            temp.next = head2;
            head2 = head2.next;
            temp = temp.next;
        }
        i++;
    }
    if (head1 != null) {
        temp.next = head1;
        head1 = head1.next;
        temp = temp.next;
    }
    dummy = dummy.next;
    return dummy;
}
```

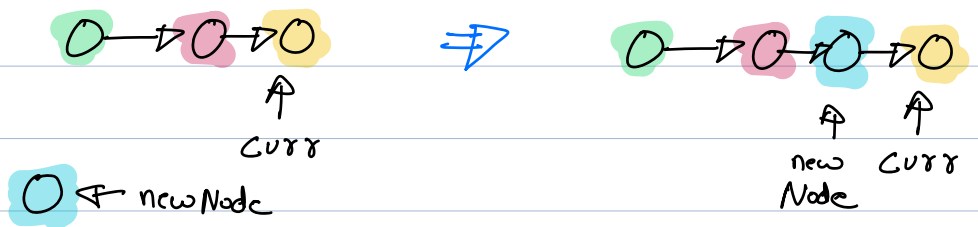
Q1 Given a reference to a node. Delete it.

Ex1



Q2 Given a pointer to a node. Insert a newnode before it.

Ex1

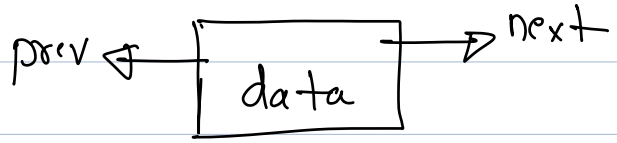


Again traverse and find 1 node before curr. TC: O(n)

Common Problem !

Can't move Backwards !

Node with a prev reference

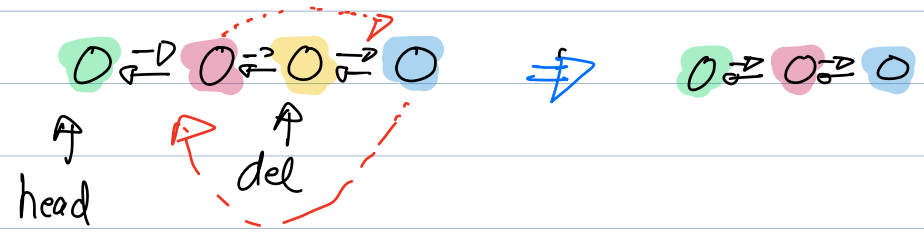



```
Class Node {
    int data;
    Node next;
    Node prev;
}
```

Doubly linked list

Q1 Given a reference to a node. Delete it.

Ex1

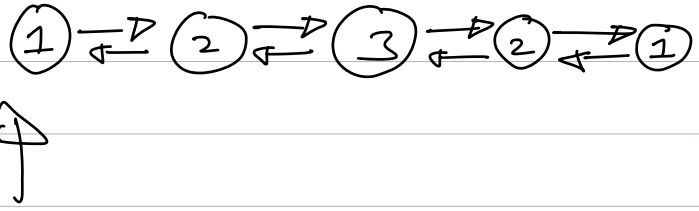



```
Node. deleteNode ( Node del , Node head) {
    if (del == null)
        return head;

    if (del == head)
        head = del.next;

    if (del.prev != null) {
        del.prev.next = del.next
    }

    if (del.next != null) {
        del.next.prev = del.prev
    }

    del.prev = null
    del.next = null.        // delete (del)

    return head;
}
```

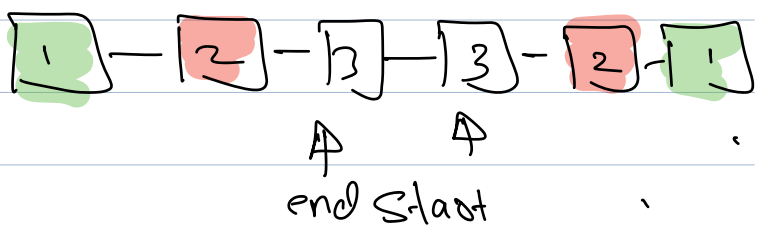
TC : O(1)

SC : O(1)

Q2 Given a DLL. Check if it is a palindrome.

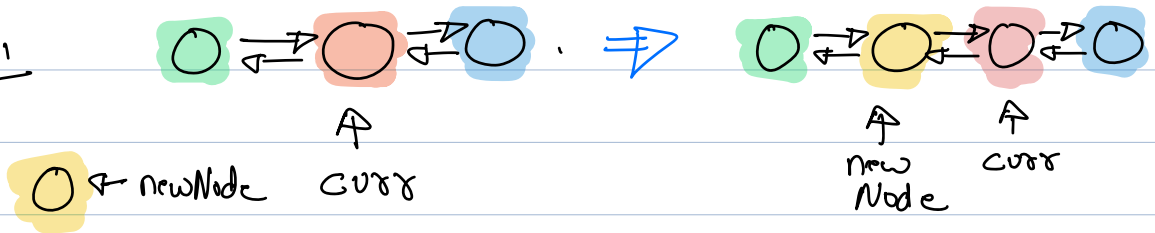



```
Node end = find Tail (head)
Node start = head.

while( start != end && end.next != start ) {

    if ( start.data != end.data)
            return false;

    start = start.next;          TC: O(n)
    end = end.prev;              SC: O(1)
}

return true.
```



Q3 Given a pointer to a node. Insert before the given node.

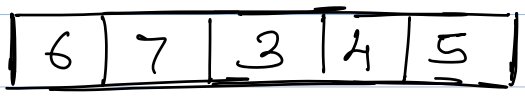
Ex1



TODO

hint

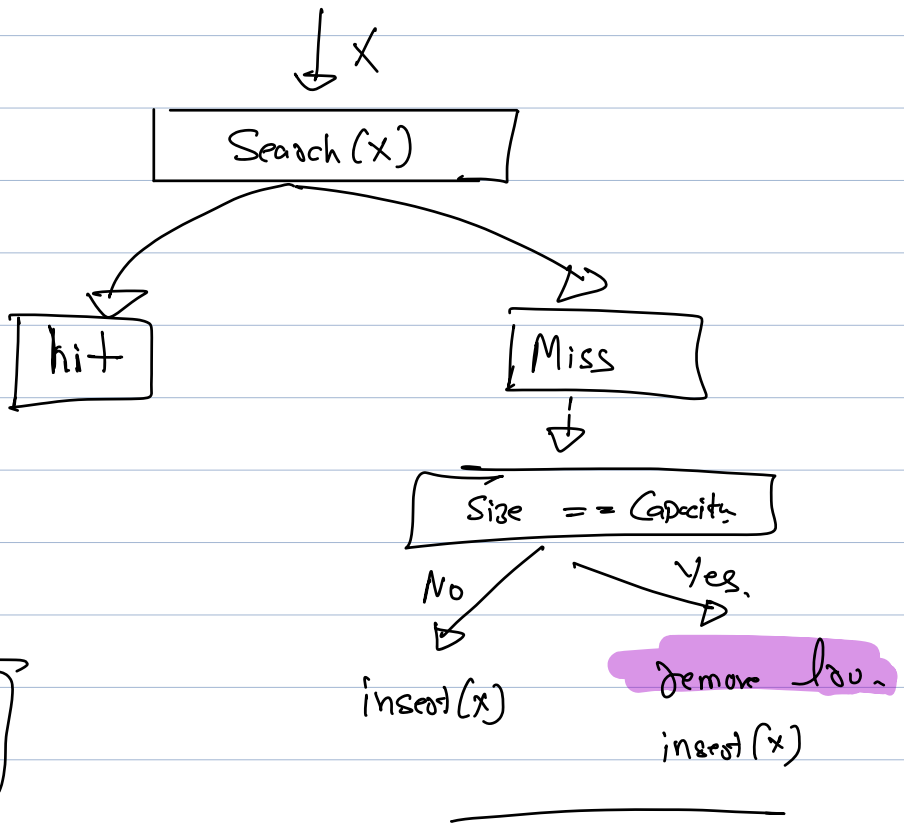
CACHE | 1, 2, 3, 3, 4, 5, 6, 7



Size = 5

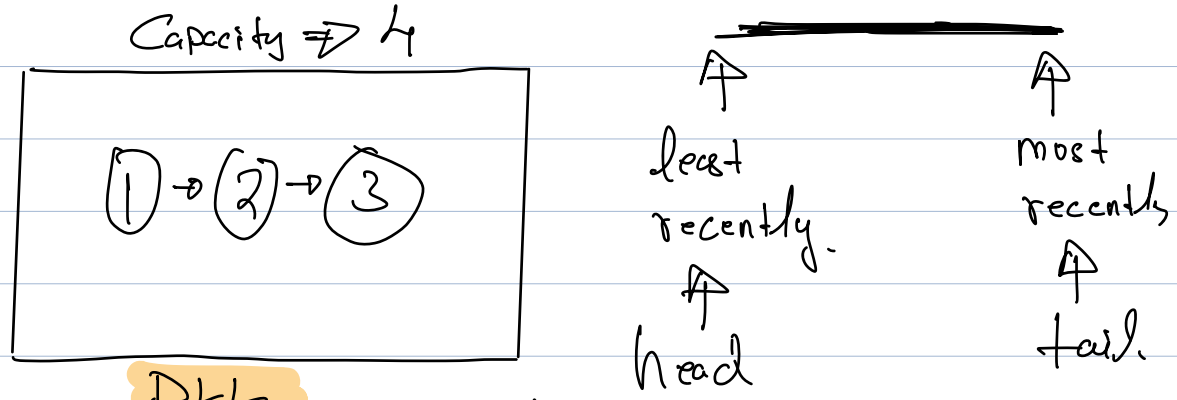
→ miss

→ hit



10:30

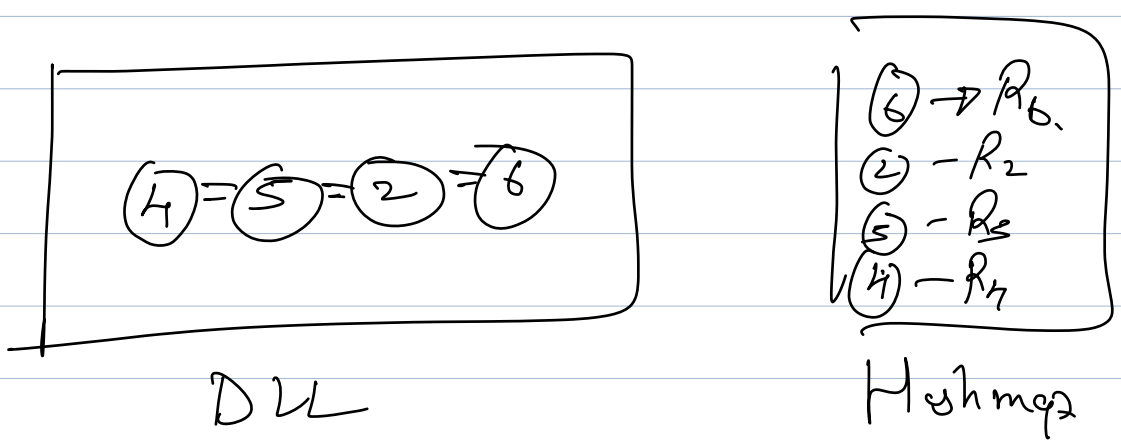
1 2 3 2 2 1 4 5 2 6



→ displace & insert at last.

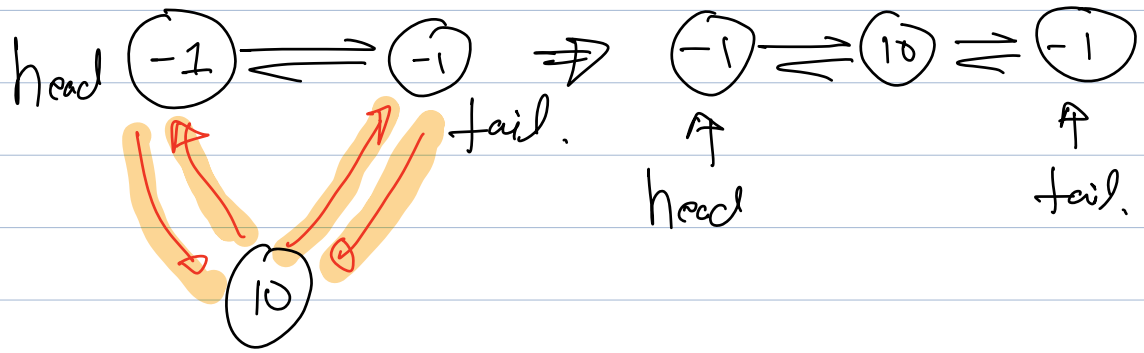
→ remove from head.

1 2 3 2 2 1 4 5 2 6

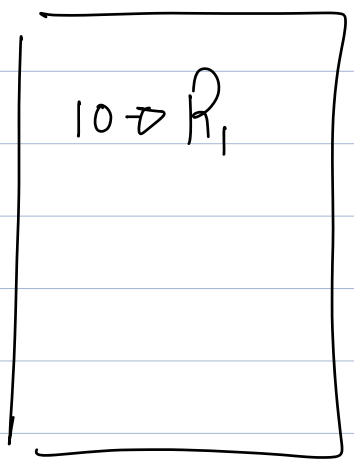


10 15 19 20 15 18 23 20 19 17 17

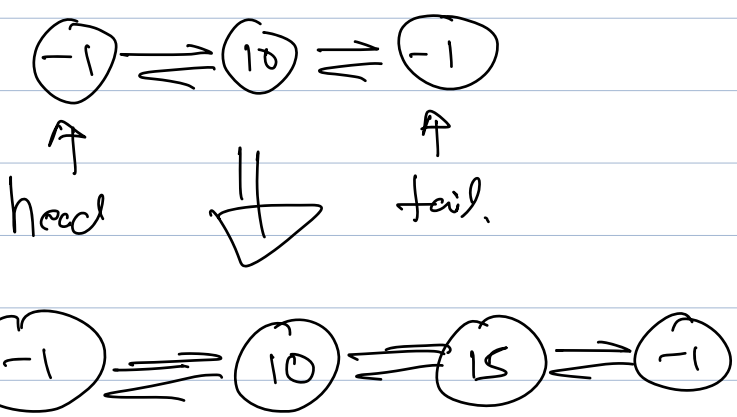
1) 10



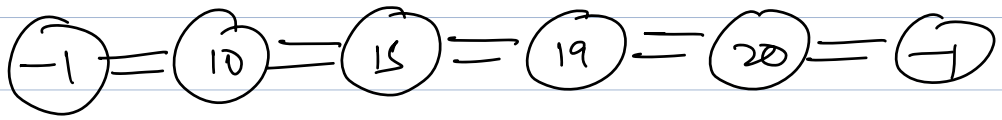

```
addToTail (Node x) {
    x.next = tail;
    tail.prev.next = x;
    x.prev = tail.prev.
    tail.prev = x;
}
```



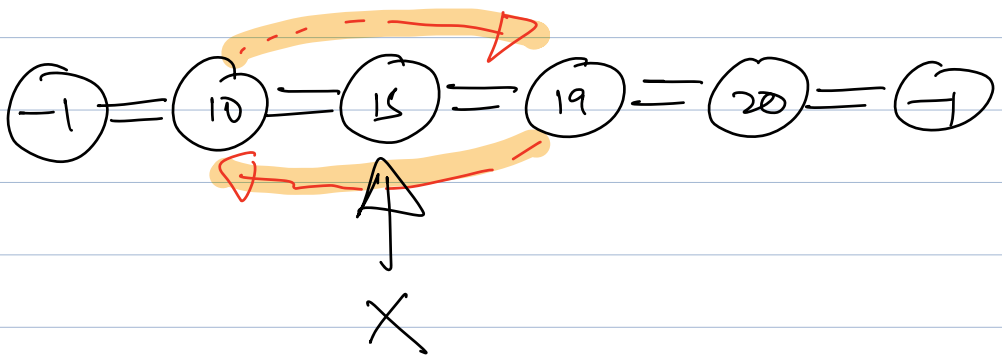
2) 15



3) 19, 20



4) 15



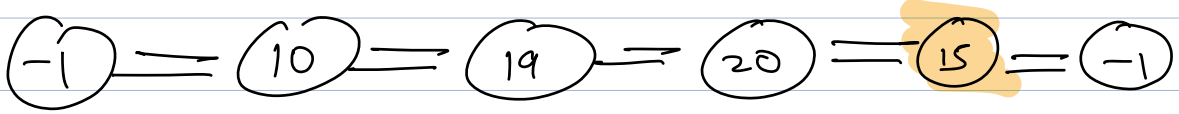
remove (Node x) {

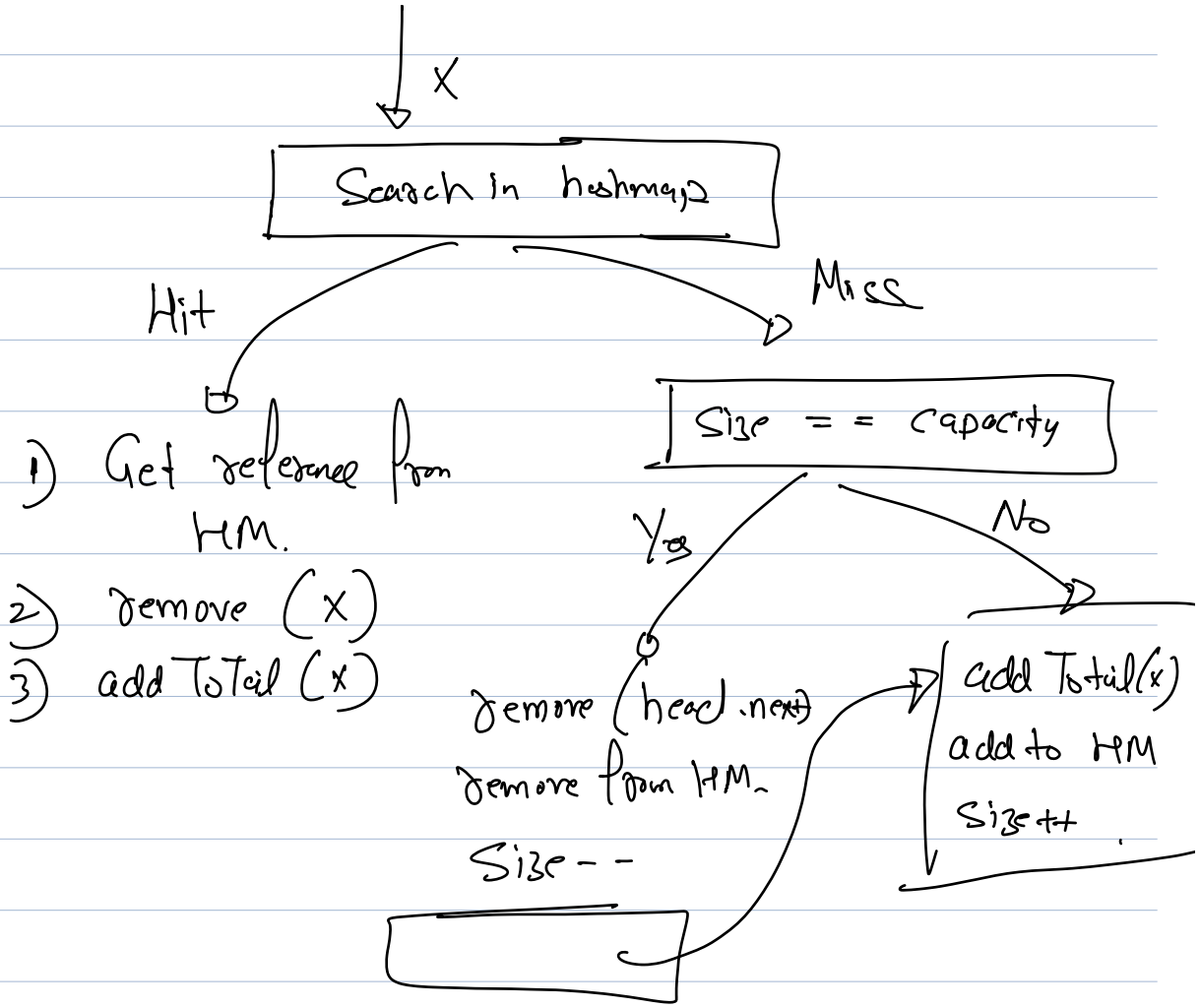
X.prev.next = x.next

X.next.prev = x.prev

}

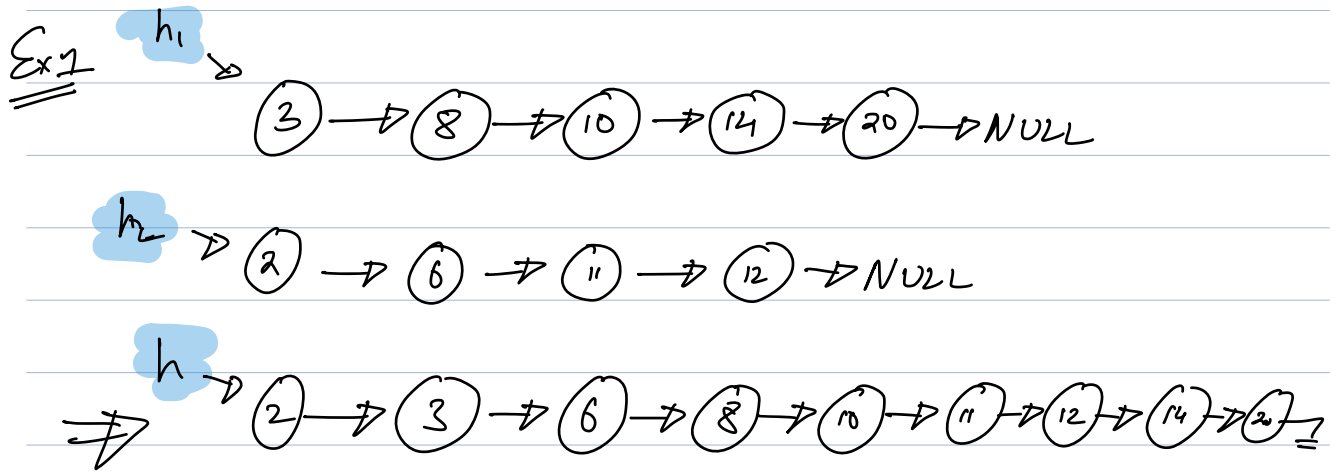
addToTail (x)




X
Search in hashmap
Hit
Miss
1) Get reference from HM.
2) remove (x)
3) add ToTail (x)
Size == capacity
Yes
No
remove (head.next)
remove from HM.
Size--
add Tail(x)
add to HM
Size++

Q Given two sorted linked list. Merge them into a single sorted LL!

Ex1



Merge Sort in linked list

Q Given a linked list. Sort it!

Ans.