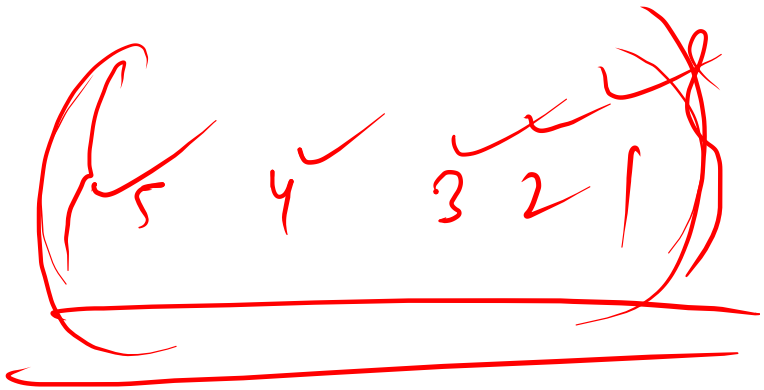
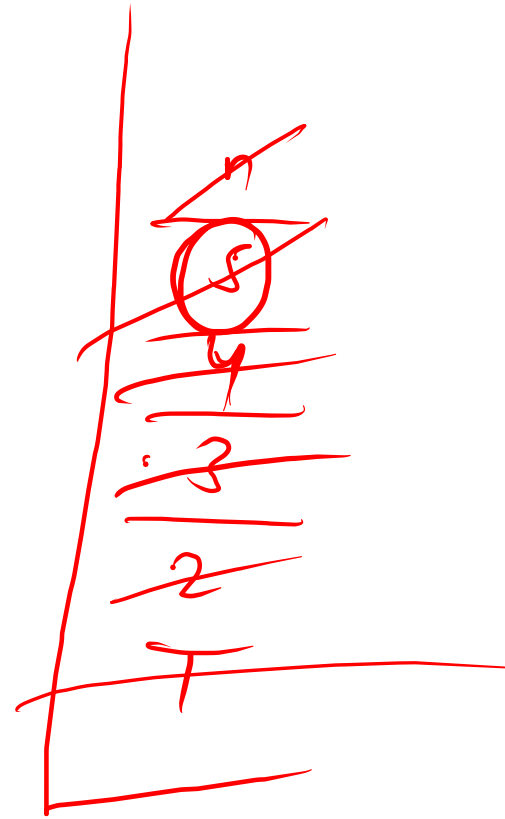
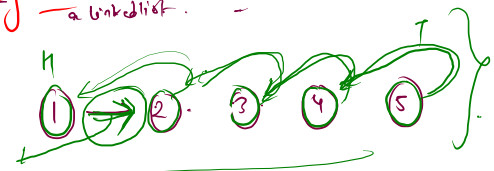
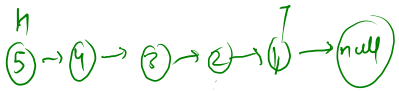

1
2
3
4
5
X


5 4 3 2 1

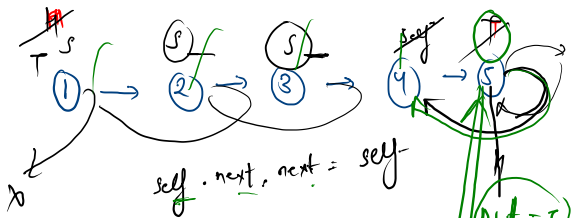

5
4
3
2
1

~~display~~ reverse. (Recursion).
a linkedlist.

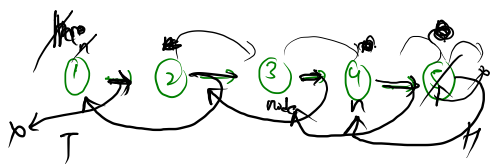




1 → null

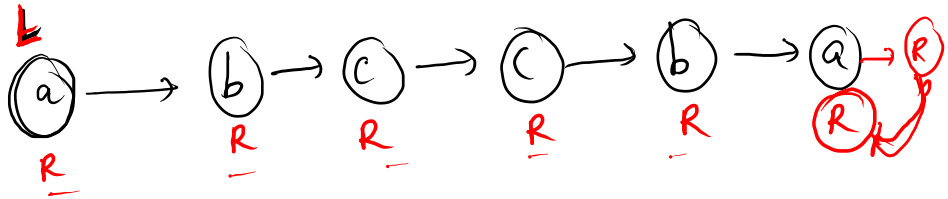


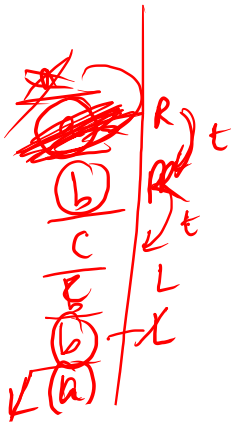

```
private void reversePRHelper(Node node){
  if(node == null)
    return;
  reversePRHelper(node.next);
  if(node != tail){
    node.next.next = node;
  }
}

public void reversePR(){
  // write your code here
  reversePRHelper(head);
  head.next = null;
  Node tmp = head;
  head = tail;
  tail = tmp;
}
```



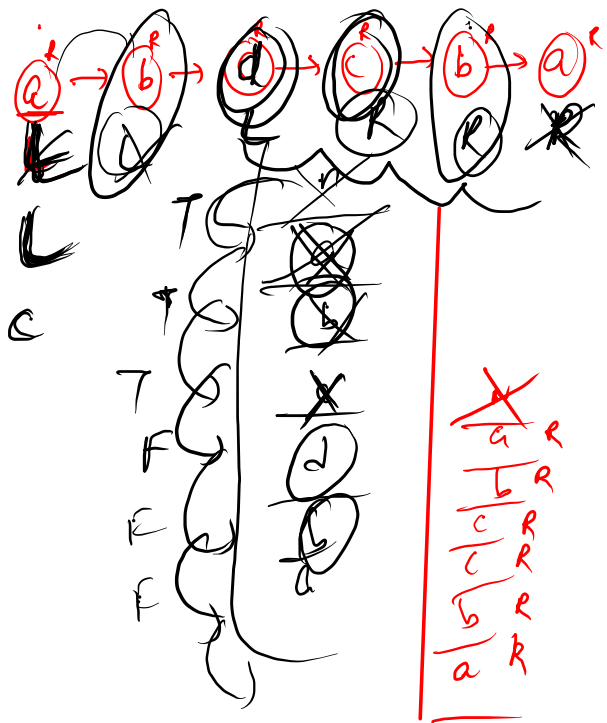
t = H
H = T
T = t

Is a LL palindrome ? (Recursion)

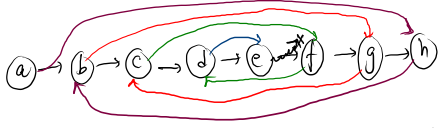


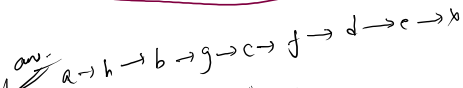


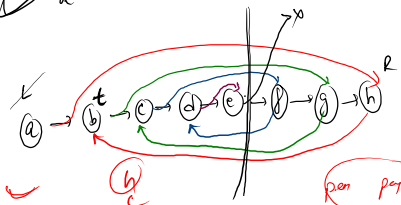
```
Node pleft;
private boolean palindromeHelper(Node right){
    if(right == null){
        return true;
    }

    boolean recAns = palindromeHelper(right.next);
    if(recAns == false){
        return false;
    }
    if(pleft.data != right.data){
        return false;
    }
    else{
        //same
        pleft = pleft.next;
        return true;
    }

}
public boolean IsPalindrome() {
  pleft = this.head;
  return palindromeHelper(head);
}
```

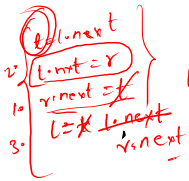


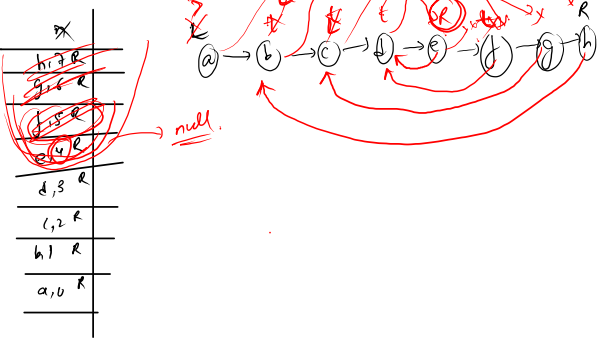
Fold.



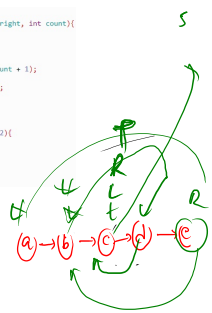








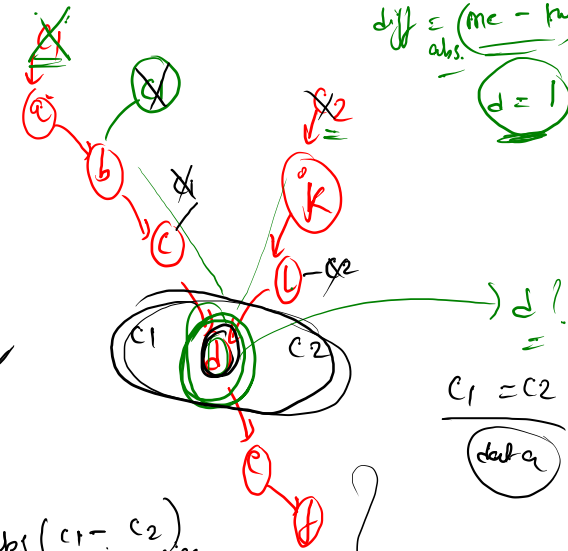
```
Node fleft;
private void foldHelper(Node right, int count){
    if(right == null){
        return;
    }

    foldHelper(right.next, count + 1);
    if(count > size / 2){
        Node temp = fleft.next;
        fleft.next = right;
        right.next = temp;
        fleft = temp;
    }
    else if( count == size / 2){
        tail = right;
        tail.next = null;
    }
}
```



Intersection.

abcdef
klmdef

one) two



diff = abs. (one - two)

d = 1

d ?

c1 = c2

data

pseudo code.

1. diff. = abs (c1.size - c2.size)
2. extra compensate.
3. travelled till we get same.
4. returned data.

clear ✓

next.

generic way.