__/__/____

Logic building in L.L

① add two numbers

- basically youre given two lists of integers in reverse order
- all you have to do is traverse through the lists till theres nodes left adding both lists nodes to a sum tracker for two nodes
- add 2 nodes
- if theres a carrory & add the sum to carry
- & keep track of carray aswell by sum/10 (keeps first part)
- now insert the sum value into a node (sum%10) keep only last part ← (13)-stored
  carry
- now just move from dummy to new node
- repeat for all nodes
- & return head.

__/__/____

② odd & even nodes

- Edge cases
  if list is empty or theres only one element return head

- initially keep the first odd node & the even node & also keep even head as first even node

- now traverse till even or evens next is null
- & ~~update odd~~ link odd to next odd
  & ~~op~~ move the odd
- same with even

- now just link the last odd to even head.

③ Sort a list of 0, 1, 2

- basically just traverse once counting 0s 1s & 2s get all its count

- traverse again from the start according to each counts overwrite those nodes with 0, 1, 2

__/__/__

(4) remove n^{th} node from back

- Edge cases
  - if head is null return null
- get the count of nodes with first traversal
- now to get n^{th} node from back you have to do (total count - n + 1) this number would be the node to remove from front.
- after getting this number
  edge cases
  - if rem is < 0 or > count then null
  - if rem is 1 then head removal.
- or else just traverse from the head while keeping a new counter when removal == count then link the prev node to current's next node & then delete current node.

__/__/____

⑤ Reverse a LL

BF — you could just use a stack store all elements of the list in the stack & then over write the LL from the start with stored stack eles

OP - basically using 3 pointer
- keep a prev node, current node & front node
- connect current node to prev
- & just move front & move prev ahead as well

LL Medium

Q Find the middle node of the list

BF - basically just count the nodes
- total nodes/2 +1 (+1 because we have to take the upper bound)
  7/2 = 3.5 so take 4
  8/2 = 4 + 1 so 5.

OP - turtle & hare method
- basically keep two pointers initially at head
- run the while loop till
  - fast->next != null - for odd lengths
  - fast != null - for even lengths

__/__/____

- ~~create the loop exits~~ and keep moving the pointers turtle by one step
hare by 2 steps
- finally just return the turtle which will be at mid.

① Detect a loop

#BF- just traverse through the list & also store each node in a map
- while traversing if you find a node in the map that's already visited then return true.

#OP- using turtle & hare method
- if there's a collision between the two pointers then it means there a loop.

② Starting point of a loop

#BF just like detecting a loop instead of returning true just return the node that gets visited the second time.

#OP- again using turtle & hare method
- but after 1st collision
- set slow to head & move both slow & fast by just one steps & the second collision point now is the start

__/__/____

③ Delete middle node

#BF - count the nodes
- Get the middle node while tracking previous
- & then just connect prev to middles next
- & delete middle.

#OP- using turtle n hare method
- get the middle node while also tracking prev
- then just connect prev to middles next
- & delete middle.