Mon Tue Wed Thu Fri Sat Sun

>Date

* Finding intersection of a L.L

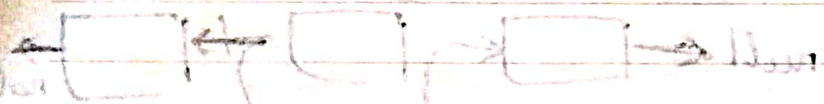
① using a list to iterate and search other list for the intersection

② using HashSet/Map on a list and search for hash value for list 2.

③ using difference in lengths.

1. find lengths of both lists.
2. move the longer list until both become equal length.
3. now move both lists together until they are ~~not~~ equal.

④ taking 2 dummy nodes and making them to point other nodes head if they reach the end.



* Adding 1 to a member represented using L.L.

① reverse L.L

② Add 1 to the initial node & carry '1' with it exceeds 9.

③ Continue until the end.

And if a carry exists till last, then add new node at end.

④ reverse the L.L for our output.

* Reversing (k-Nodes Group) for whole L.L.

Mon Tue Wed Thu Fri Sat Sun

>Date

We 1st need to find 1st k Nodes, then reverse it, then arrange it back in the original L.L.

Hence, we use reverse (L.L)
finding the next Node after the k^{th} node
function to do all these.

```
Node reverse (Node head)
{
    Node prev=null, temp=head;
    while( temp!=null)
    {
        Node nextNode= temp.next;
        temp.next= prev;
        prev= temp;
        temp= nextNode;
    }
    return prev;
}

Node findkth (Node head, int k)
{ we need to return the kth node from head
    if available else null;

    Node temp= head;
    while (--k>0 && temp!=null)
        temp= temp.next;
    return temp.
```