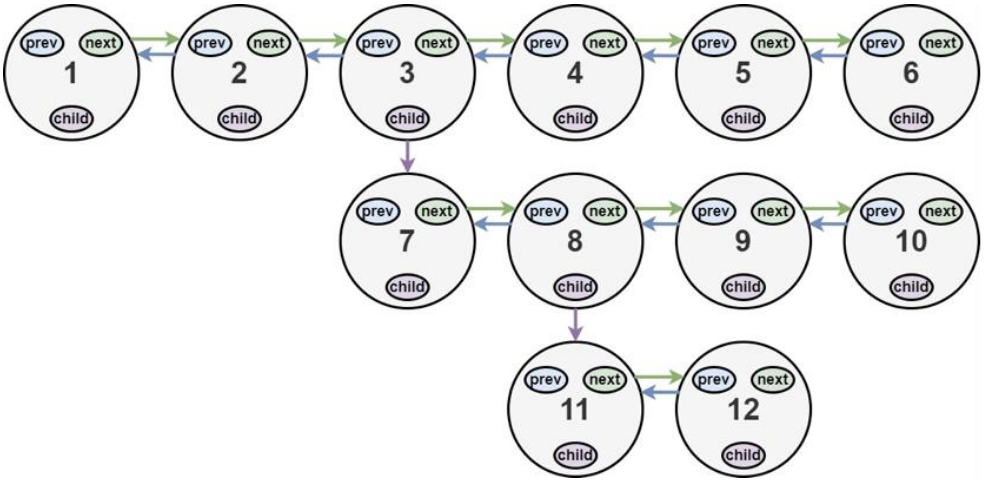
1. You are given a doubly linked list, which contains nodes that have a next pointer, a previous pointer, and an additional child pointer. This child pointer may or may not point to a separate doubly linked list, also containing these special nodes. These child lists may have one or more children of their own, and so on, to produce a multilevel data structure as shown in the example below.

   Given the head of the first level of the list, flatten the list so that all the nodes appear in a single-level, doubly linked list. Let curr be a node with a child list. The nodes in the child list should appear after curr and before curr.next in the flattened list.

   Return the head of the flattened list. The nodes in the list must have all of their child pointers set to null.

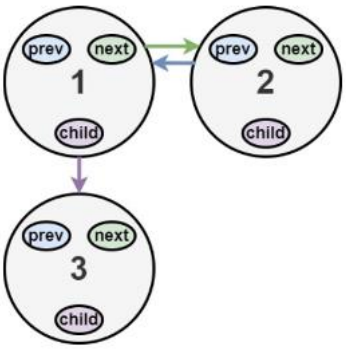
   Example 1:



   Input: head = [1,2,3,4,5,6,null,null,null,7,8,9,10,null,null,11,12]

   Output: [1,2,3,7,8,11,12,9,10,4,5,6]

   Example 2:



   Input: head = [1,2,null,3]

Output: [1,3,2]

Tips:

i. Start form the head, move one step each time to the next node
ii. When meet with a node with child, say node p, follow its child chain to the end and connect the tail node with p.next, by doing this we merged the child chain back to the main thread
iii. Return to p and proceed until you find the next node with child.
iv. Repeat until reach null

```
/*
// Definition for a Node.
class Node {
    public int val;
    public Node prev;
    public Node next;
    public Node child;
};
*/


class Solution {
    public Node flatten(Node head) {


    }
}
```