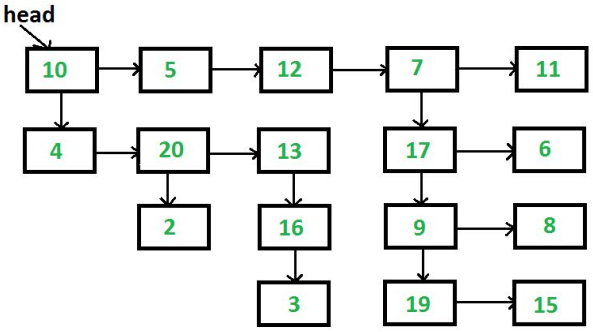
Task 3a: Given a linked list where in addition to the next pointer, each node has a child pointer, which may or may not point to a separate list. These child lists may have one or more children of their own, and so on, to produce a multilevel data structure, as shown in below figure.You are given the head of the first level of the list. Flatten the list so that all the nodes appear in a single-level linked list. You need to flatten the list in way that head comes first then children then next. Assume that the node struct looks something like this.

```
struct List
{
    int data;
    struct List *next;
    struct List *child;
};
```



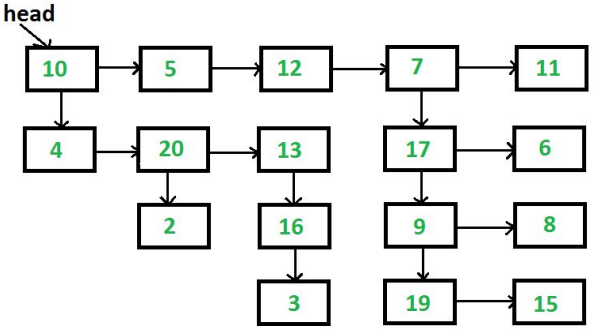
Example: The above list should be converted to
10->4->20->2->13->16->3->5->12->7->17->9->19->8->->6->11
Possible Implementation: Recursively perform depth first search and adjust pointers such that when you reach the end of all the subchildren of curr the last node's next is set to curr.next, curr.next is set to curr.child, and curr.child is set to null.

Task 3b: Given a singly linked list, group all odd nodes together followed by the even nodes. Please note here we are talking about the node number and not the value in the nodes.
Example:
Given 1->2->3->4->5->NULL,
return 1->3->5->2->4->NULL.
Possible Implementation: Create node pointers odd, even, and evenHead. Keep iterating through the list setting odd.next = odd.next.next; even.next = even.next.next; odd = odd.next; even = even.next. At the end set odd.next = evenHead. Tip: try drawing out some examples to check for understanding.

Task 3a: Given a linked list where in addition to the next pointer, each node has a child pointer, which may or may not point to a separate list. These child lists may have one or more children of their own, and so on, to produce a multilevel data structure, as shown in below figure.You are given the head of the first level of the list. Flatten the list so that all the nodes appear in a single-level linked list. You need to flatten the list in way that head comes first then children then next. Assume that the node struct looks something like this.

```
struct List
{
    int data;
    struct List *next;
    struct List *child;
};
```

Example: The above list should be converted to
10->4->20->2->13->16->3->5->12->7->17->9->19->8->->6->11
Possible Implementation: Recursively perform depth first search and adjust pointers such that when you reach the end of all the subchildren of curr the last node's next is set to curr.next, curr.next is set to curr.child, and curr.child is set to null.



Task 3b: Given a singly linked list, group all odd nodes together followed by the even nodes. Please note here we are talking about the node number and not the value in the nodes.
Example:
Given 1->2->3->4->5->NULL,
return 1->3->5->2->4->NULL.
Possible Implementation: Create node pointers odd, even, and evenHead. Keep iterating through the list setting odd.next = odd.next.next; even.next = even.next.next; odd = odd.next; even = even.next. At the end set odd.next = evenHead. Tip: try drawing out some examples to check for understanding.