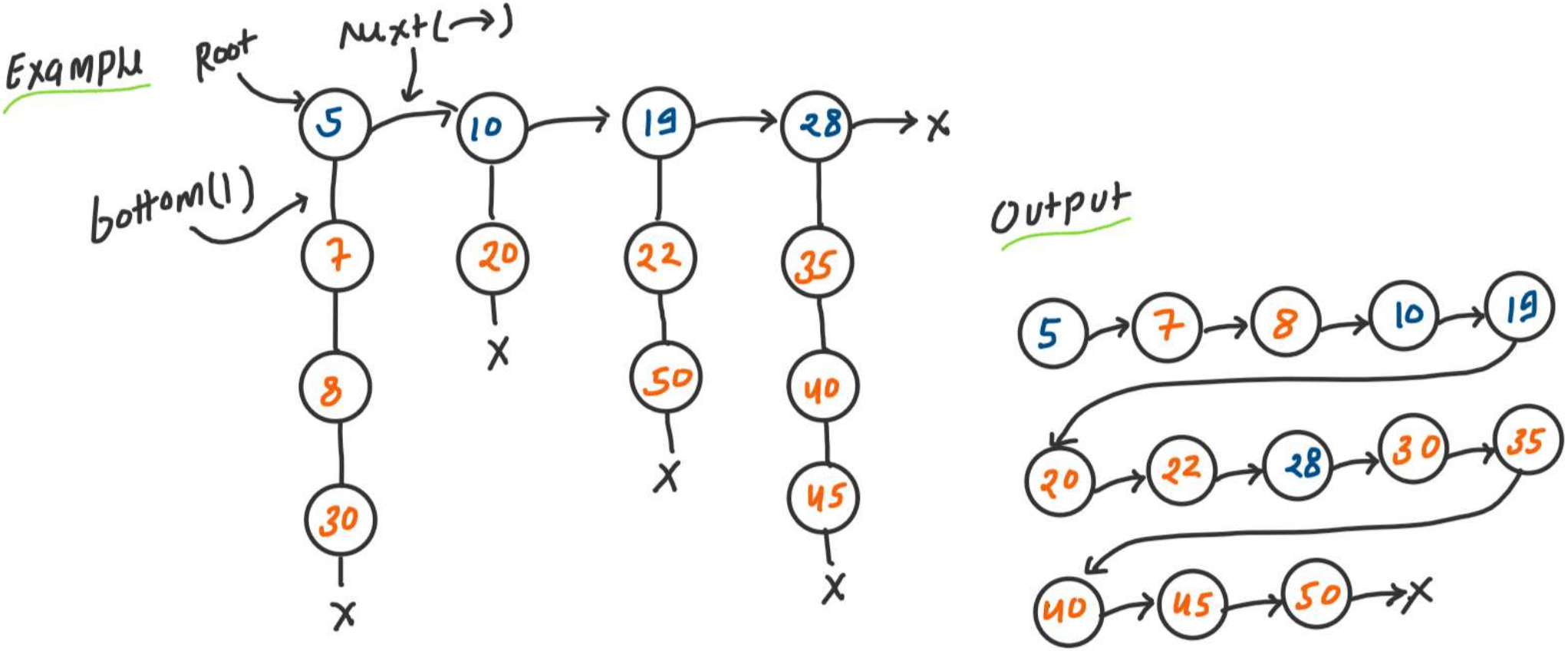
HW 06: Flatten Linked List (GFG)

PROBLEM STATEMENT:
Given a Linked List of size N, where every node represents a ***sub-linked-list*** *and contains two pointers:*
(I) a ***next*** *pointer to the next node,*
(II) a ***bottom*** *pointer to a linked list where this node is head.*

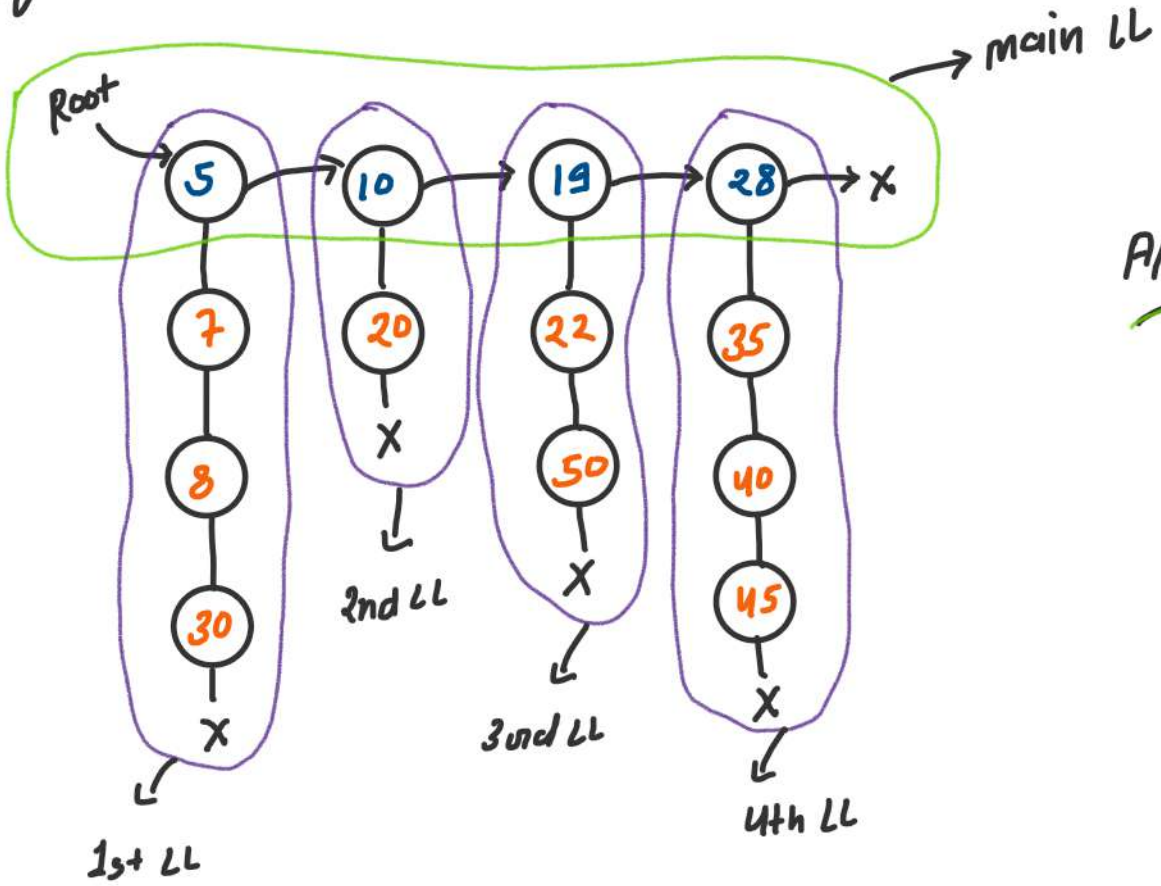
Each of the sub-linked-list is in sorted order.

Flatten the Link List such that all the nodes appear in a single level while maintaining the sorted order.

Note: *The flattened list will be printed using the bottom pointer instead of the next pointer.*


Example
Root
Next(→)
bottom(↓)
5
10
19
28
X
7
20
22
35
8
50
40
30
45
X
X
X
X
Output
5
7
8
10
19
20
22
28
30
35
40
45
50
X

Build Logic

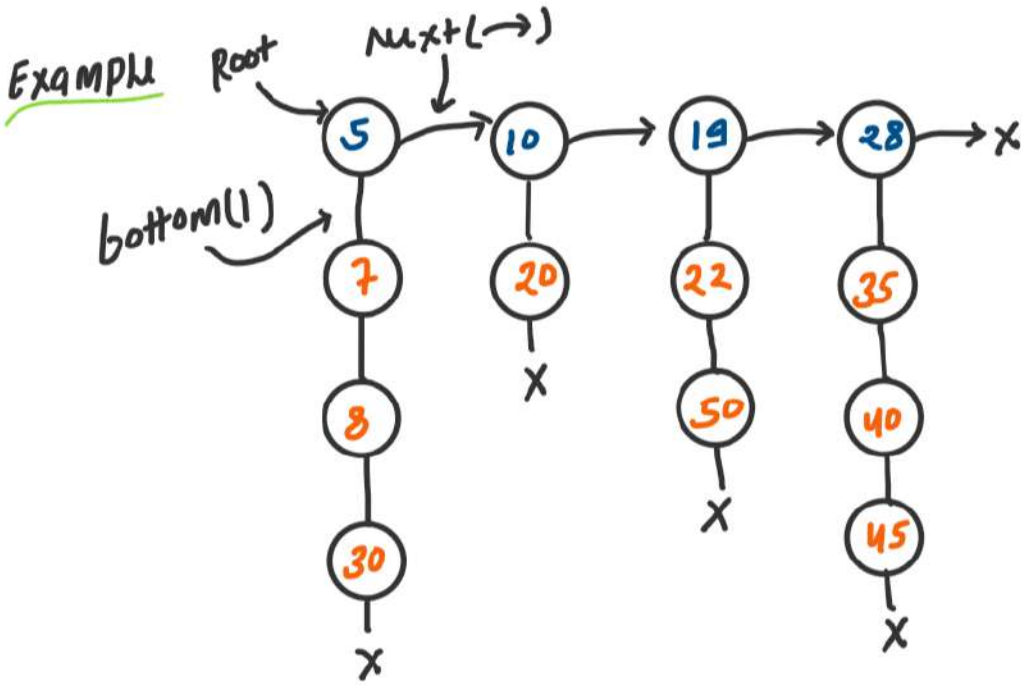


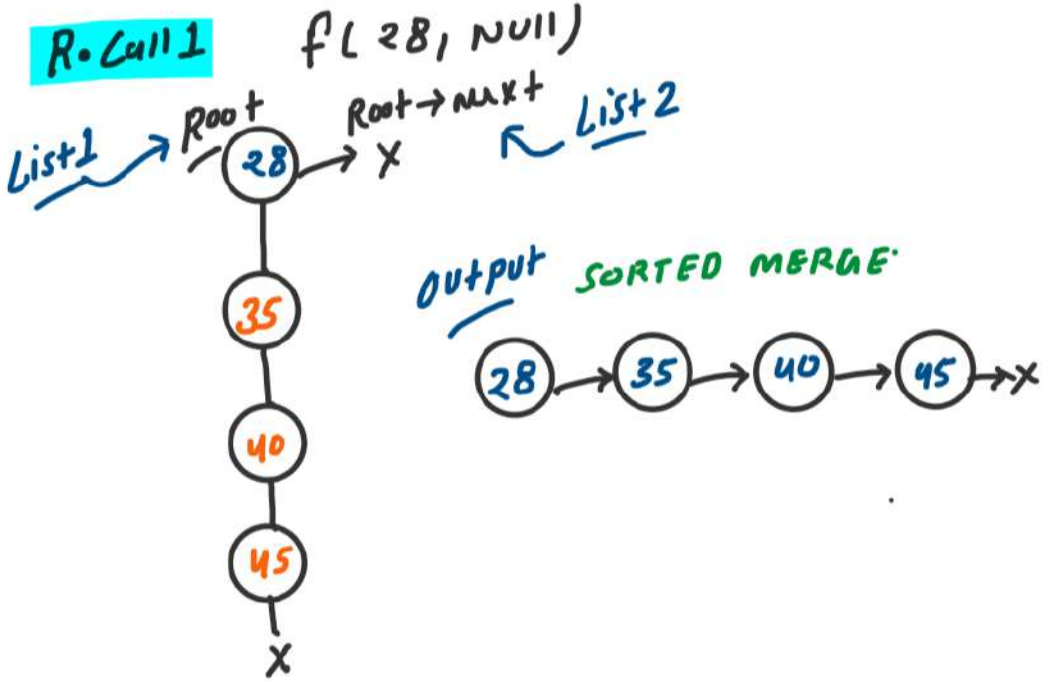
Approach: Recursive Approach

Step1 Merge Two sorted LL

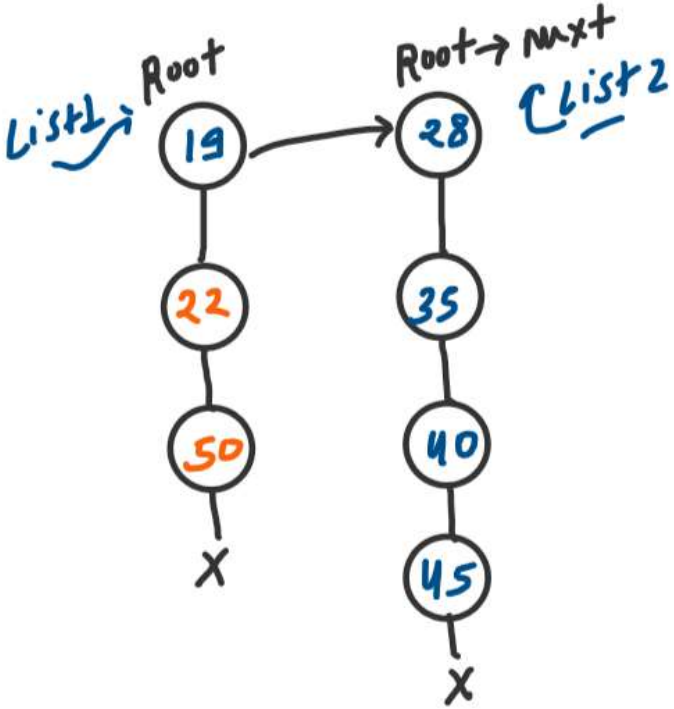
Step2 Merge a next sorted LL with already two merged LL

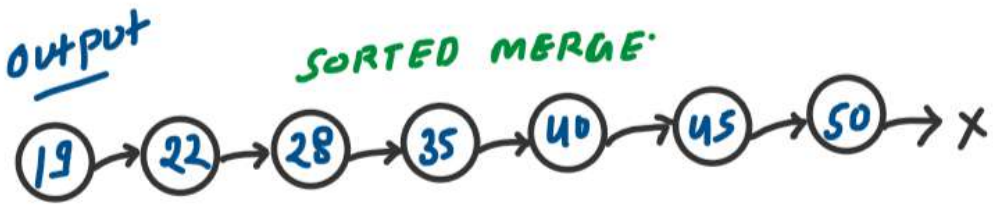
DRY RUN


Example
Root
Next(→)
bottom(1)
5
10
19
28
X
7
20
22
35
8
50
40
30
45
X
X
X
X

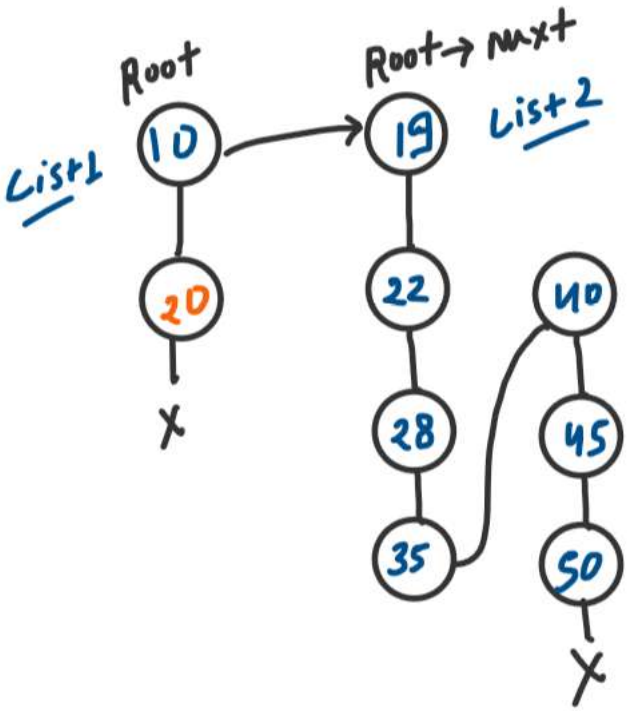

R. Call 1
f(28, NULL)
List1
Root
Root→Next
List2
28
X
35
40
45
X
Output
SORTED MERGE
28
35
40
45
X

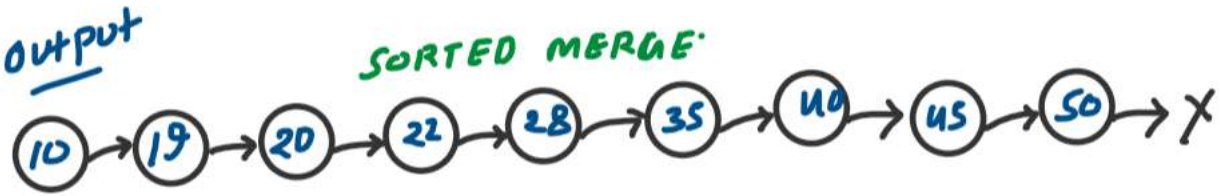
R. Call 2 f(19, 28)



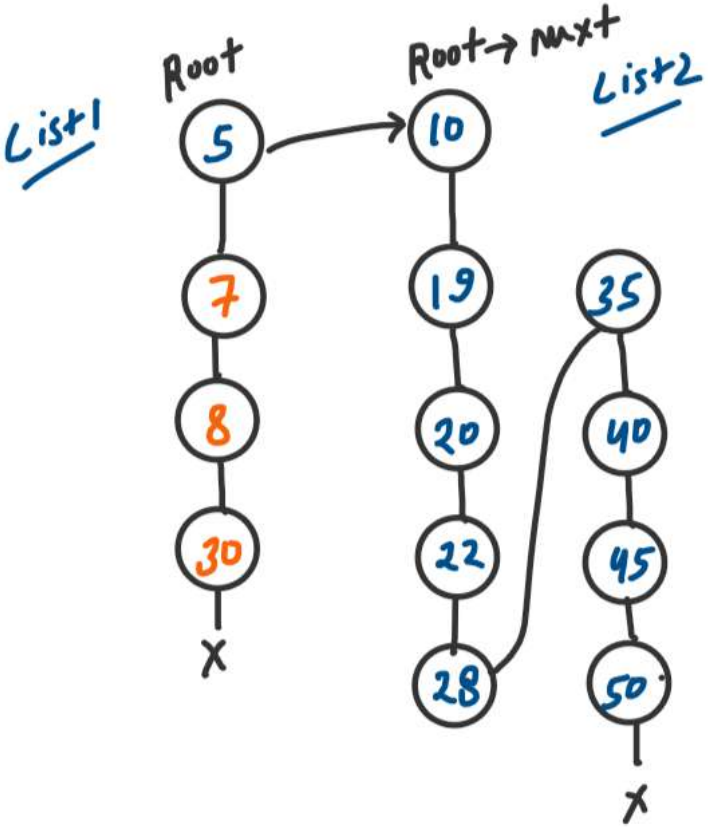


R.Call 3 fL(10, 19)



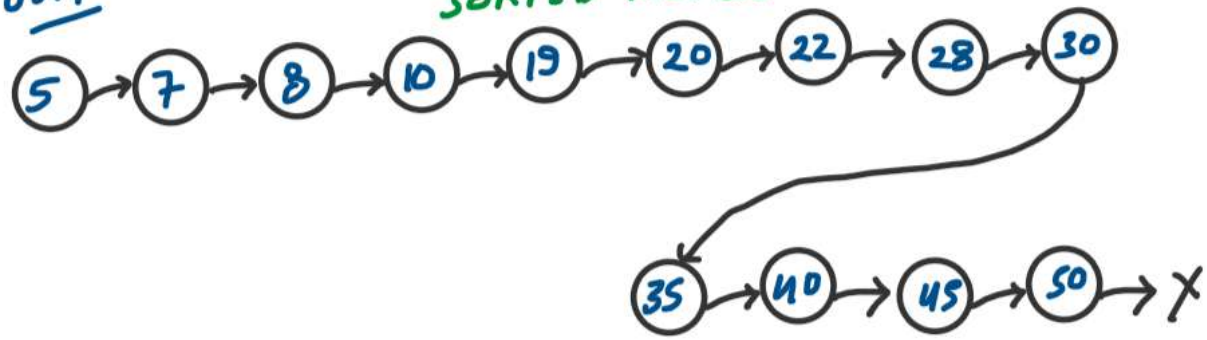


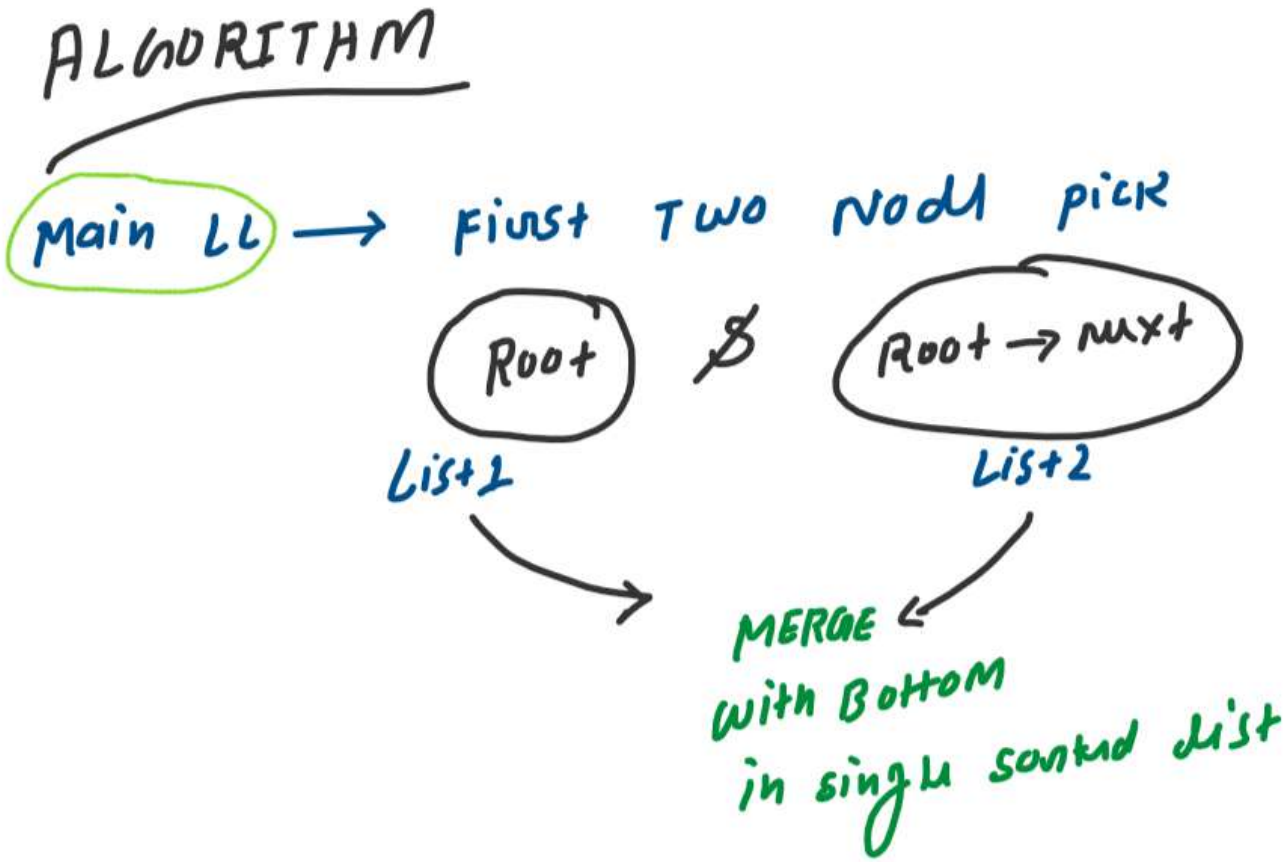
R. Call U f(5 , 10)



Final output

SORTED MERGE:




ALGORITHM
Main LL
First Two Node pick
Root
&
Root → next
List1
List2
MERGE
with Bottom
in single sorted list

```
/* Node structure  used in the program
struct Node{
    int data;
    struct Node * next;
    struct Node * bottom;

    Node(int x){
        data = x;
        next = NULL;
        bottom = NULL;
    }
};
*/

Node* mergeTwoSortedLL(Node* list1, Node* list2){...}

Node *flatten(Node *root)
{
   // Base case
   if(root == NULL) return NULL;

   Node* mergedLL = mergeTwoSortedLL(root, flatten(root->next));
   return mergedLL;
}
```

```
Node* mergeTwoSortedLL(Node* list1, Node* list2){
    // Base case
    if(list1 == NULL) return list2;
    if(list2 == NULL) return list1;

    // Processing
    Node* ans = NULL;
    if(list1->data < list2->data){
        ans = list1;
        list1->bottom = mergeTwoSortedLL(list1->bottom, list2);
    }
    else{
        ans = list2;
        list2->bottom = mergeTwoSortedLL(list1, list2->bottom);
    }

    return ans;
}
```