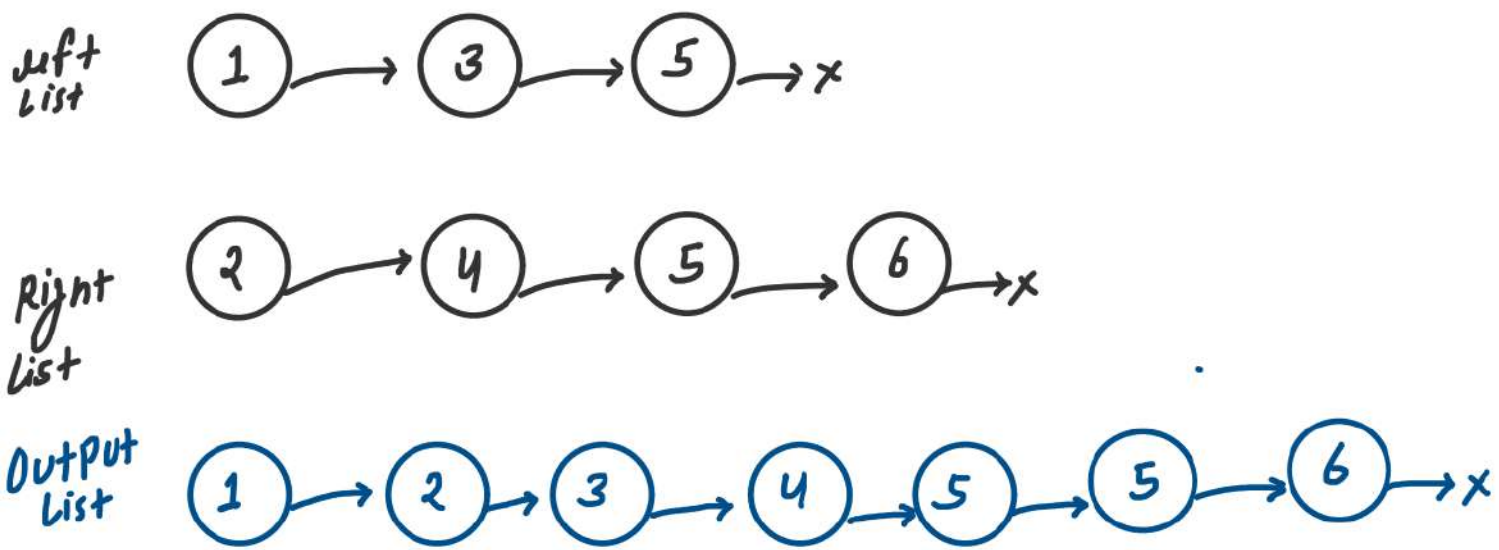
HW 01: Merge Two Sorted Lists (Leetcode-21)

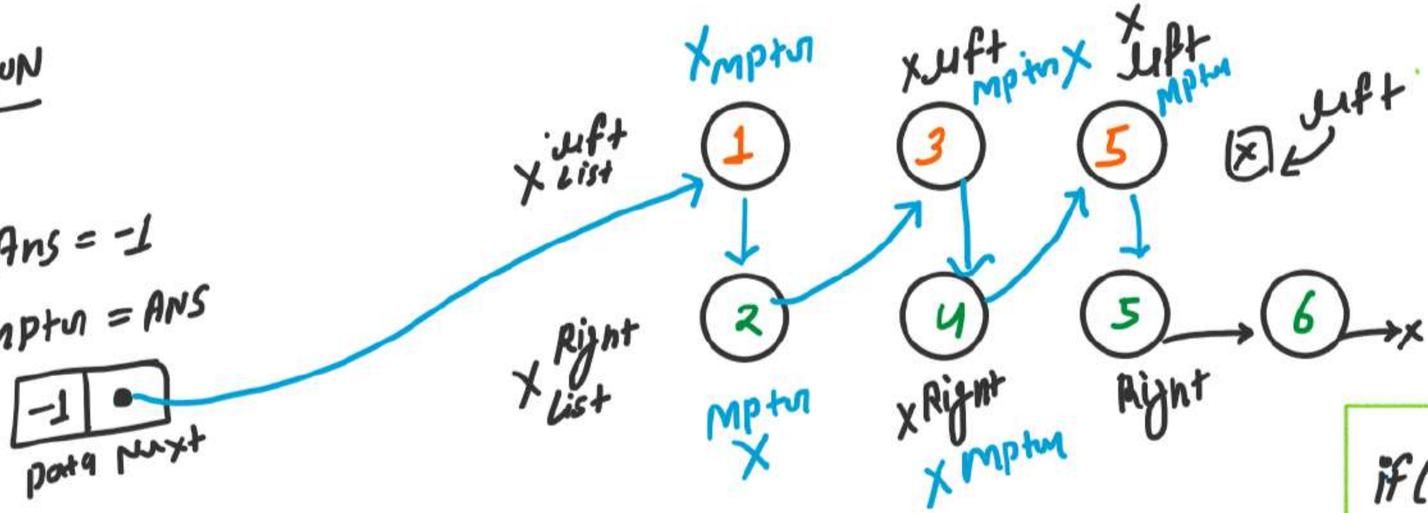
Ex



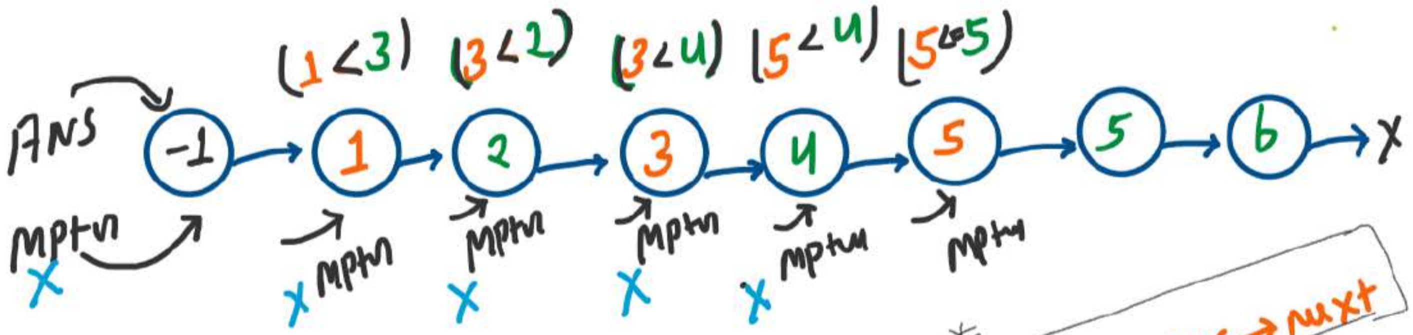
DRY RUN

let Ans = -1

mptr = ANS




```
if(left){
    mptr->next = left;
}
if(Right){
    mptr->next = Right;
}
```




```
if(left->data <= right->data)
{
    mptr->next = left;
    mptr = left;
    left = left->next;
}
Else{
    mptr->next = right;
    mptr = Right;
    Right = Right->next;
}
```

** return Ans->next

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* left, ListNode* right) {
        if(left == NULL) return right;
        if(right == NULL) return left;

        ListNode* ans = new ListNode(-1);
        ListNode* mptr = ans;

        while(left != NULL && right != NULL){
            if(left->val <= right->val){
                mptr->next = left;
                mptr = left;
                left = left->next;
            }
            else{
                mptr->next = right;
                mptr = right;
                right = right->next;
            }
        }

        if(left != NULL){
            mptr->next = left;
            // mptr = left;
            // left = left->next;
        }

        if(right != NULL){
            mptr->next = right;
            // mptr = right;
            // right = right->next;
        }

        return ans->next;
    }
};
```

Time complexity = O(N)

where, N is total numbers of Nodes of both list.

Space complexity = O(1)

No extra space used by ANS and mptrs.