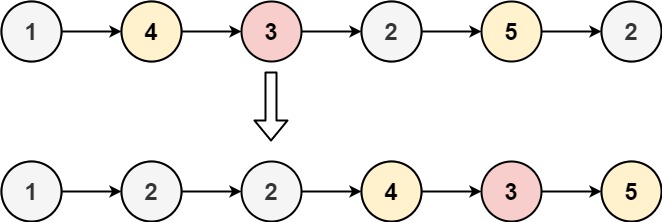
**86. Partition List :-**

Medium Accepted: 484.8K Submissions: 911.1K Acceptance Rate: 53.2%

Given the head of a linked list and a value x, partition it such that all nodes **less than** x come before nodes **greater than or equal** to x.

You should **preserve** the original relative order of the nodes in each of the two partitions.

**Example 1:**



**Input:** head = [1,4,3,2,5,2], x = 3

**Output:** [1,2,2,4,3,5]

**Example 2:**

**Input:** head = [2,1], x = 2

**Output:** [1,2]

**Constraints:**

* The number of nodes in the list is in the range [0, 200].
* -100 <= Node.val <= 100
* -200 <= x <= 200

**Code :-**

/\*\*

 \* 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\* partition(ListNode\* head, int x) {

        if(!head || !head->next)     return head;

        queue<int> q;

        ListNode \*ptr=head, \*small=head;

        while(ptr){

            if(ptr->val < x){

                small->val = ptr->val;

                small = small -> next;

            }

            else

                q.push(ptr->val);

            ptr = ptr->next;

        }

        ptr = small;

        while(q.size()){

            ptr->val = q.front();

            q.pop();

            ptr = ptr->next;

        }

        return head;

    }

};

**T.C :- O(N)**

**S.C :- O(N)**