**Problem Name:** Partition list

**Topics:** Linked list, Two Pointers

**Companies:** Amazon, Microsoft, Apple, Facebook, Bloomberg

**Level:** Medium

**Language:** C++

**Problem Statement**: 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.

**Input Format:**

First line of the input contain integer n (size of list)

Second line contain n space separated integer list values.

Last line contain integer value x.

Ex:

5

1 2 3 4 5

3

**Output Format:** Print the resultant linked list

**Constraints:**

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

**Examples:**

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

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

**Solution:**

**Explanation:**   Traverse the list and swap the last node with current node if current node is smaller than x and last node node is greater than current node .

**Code:**

#include <bits/stdc++.h>

using namespace std;

class ListNode

{

    public:

        int val;

        ListNode\* next;

        ListNode(int a){

            val = a;

            next = NULL;

        }

};

void insertNode(ListNode\* &head,int val) {

    ListNode\* newNode = new ListNode(val);

    if(head == NULL) {

        head = newNode;

        return;

    }

    ListNode\* temp = head;

    while(temp->next != NULL)

     temp = temp->next;

    temp->next = newNode;

    return;

}

void printList(ListNode \*node)

{

    while (node!=NULL)

    {

        cout<<node->val<<" ";

        node = node->next;

    }

}

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

    if(!head)

        return NULL;

    ListNode\* p, \*q,\*r,\*t;

    p=head;

    q=head->next;

    r=NULL;

    bool b=true;

    // exit from loop if no change posiible so b became false

    while(b)

    {

        b=false;

            p=head;

    q=head->next;

    r=NULL;

            // cuurent senorio r p q

        // in below while loop we swap current and last node

        while(q)

        {

            if(q->val<x && (p->val>=x))

            {  if(head==p )

                head=q;

                if(r)

                r->next=q;

                if(p)

                p->next=q->next;

                q->next=p;

                // after swapping  the order became r q p  so swap p and q

                // swapping of p and q

                t=p;

                p=q;

                q=t;

                b=true;

            }

            // cuurent senorio r p q

                r=p;

                p=q;

                q=q->next;

            // now move forward r become p ,p becomes q and q becomes its next

        }

    }

    return head;

}

int main() {

    ListNode\* a = NULL;

    ListNode\* result =NULL;

    int n, temp, x;

    cin>>n;

    while(n--){

        cin>>temp;

        insertNode(a, temp);

    }

    cin>>x;

    result = partition(a, x);

    printList(result);

    return 0;

}

**Time Complexity**: O(N)

**Space Complexity:** O(1)

**Optimized Solution:**

**Explanation:** the basic idea is to maintain two queues, the first one stores all nodes with val less than x , and the second queue stores all the rest nodes. Then concat these two queues. Remember to set the tail of second queue a null next.

**Code:**

#include <bits/stdc++.h>

using namespace std;

class ListNode

{

    public:

        int val;

        ListNode\* next;

        ListNode(int a){

            val = a;

            next = NULL;

        }

};

void insertNode(ListNode\* &head,int val) {

    ListNode\* newNode = new ListNode(val);

    if(head == NULL) {

        head = newNode;

        return;

    }

    ListNode\* temp = head;

    while(temp->next != NULL)

     temp = temp->next;

    temp->next = newNode;

    return;

}

void printList(ListNode \*node)

{

    while (node!=NULL)

    {

        cout<<node->val<<" ";

        node = node->next;

    }

}

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

    ListNode left(0), right(0);

    ListNode \*l = &left, \*r = &right;

    while(head){

        ListNode\* & ref = head->val < x ? l : r;

        ref->next = head;

        ref = ref->next;

        head = head->next;

    }

    l->next = right.next;

    r->next = NULL;

    return left.next;

}

int main() {

    ListNode\* a = NULL;

    ListNode\* result =NULL;

    int n, temp, x;

    cin>>n;

    while(n--){

        cin>>temp;

        insertNode(a, temp);

    }

    cin>>x;

    result = partition(a, x);

    printList(result);

    return 0;

}

**Time Complexity**: O(1)

**Space Complexity:** O(1)