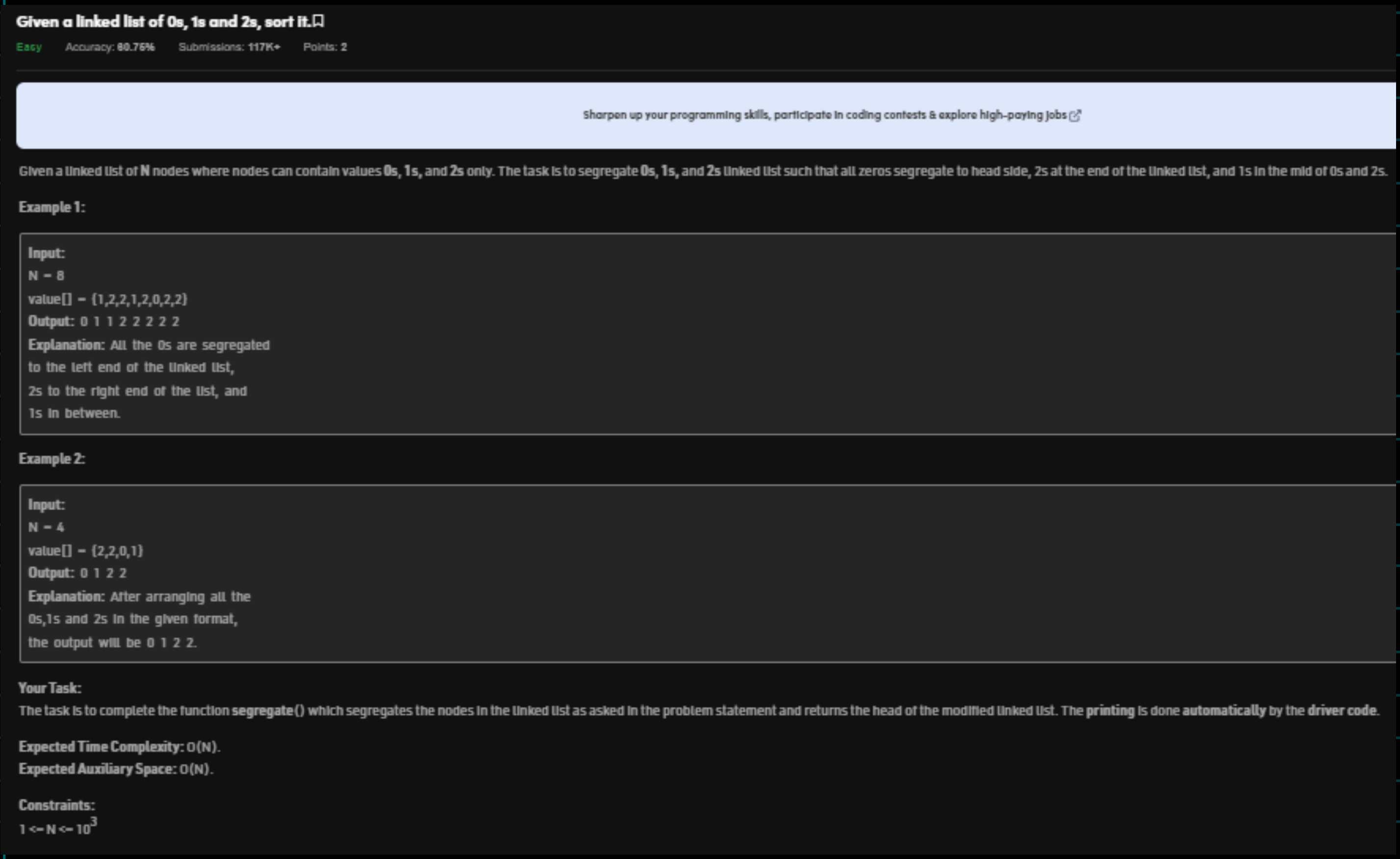
Sort 0s,1s,2s in linked list



Given a linked list of 0s, 1s and 2s, sort it.

Easy Accuracy: 60.75% Submissions: 117K+ Points: 2

Sharpen up your programming skills, participate in coding contests & explore high-paying jobs

Given a linked list of **N** nodes where nodes can contain values **0s, 1s,** and **2s** only. The task is to segregate **0s, 1s,** and **2s** linked list such that all zeros segregate to head side, 2s at the end of the linked list, and 1s in the mid of 0s and 2s.

Example 1:

```
Input:
N = 8
value[] = {1,2,2,1,2,0,2,2}
Output: 0 1 1 2 2 2 2 2
Explanation: All the 0s are segregated
to the left end of the linked list,
2s to the right end of the list, and
1s in between.
```

Example 2:

```
Input:
N = 4
value[] = {2,2,0,1}
Output: 0 1 2 2
Explanation: After arranging all the
0s,1s and 2s in the given format,
the output will be 0 1 2 2.
```

Your Task:
The task is to complete the function **segregate()** which segregates the nodes in the linked list as asked in the problem statement and returns the head of the modified linked list. The **printing** is done **automatically** by the **driver code**.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(N).

Constraints:
$1 <= N <= 10^3$

Approch 1:

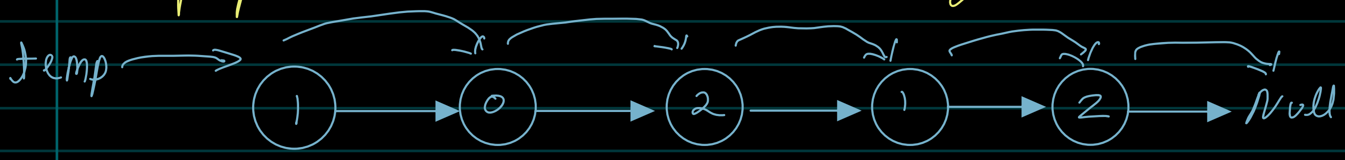
Since we are given with 0, 1 and 2 we can create 3 variable which will store count of 0, 1 and 2 and we can run 3 loop to manually override array

[2, 0, 2, 1, 1, 0] $\longrightarrow$

count1 = 2
count2 = 2
count0 = 2

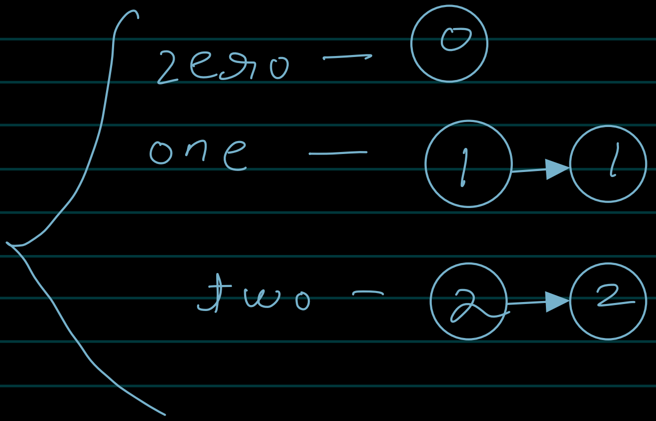
T.C – O(n)
S.C – O(1)

Approch 2 : Change Pointer

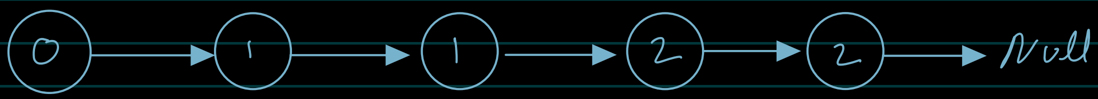


Create linked list for 0s, 1s & 2s, and traver over given linked List

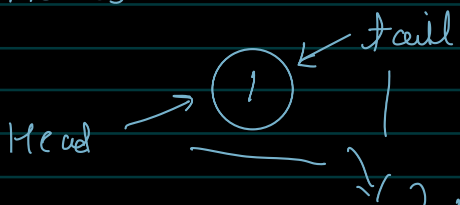
i.e



Merge all above linked list

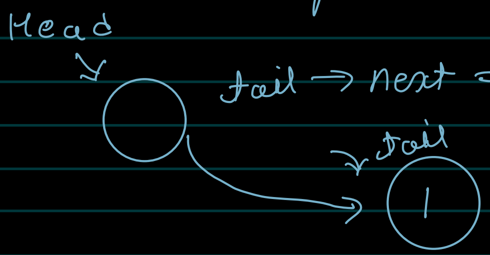


To avoid if else conditions we will be using dummy nodes



we wouldn't pointer to the head (Merge Karne mai asani hogi)

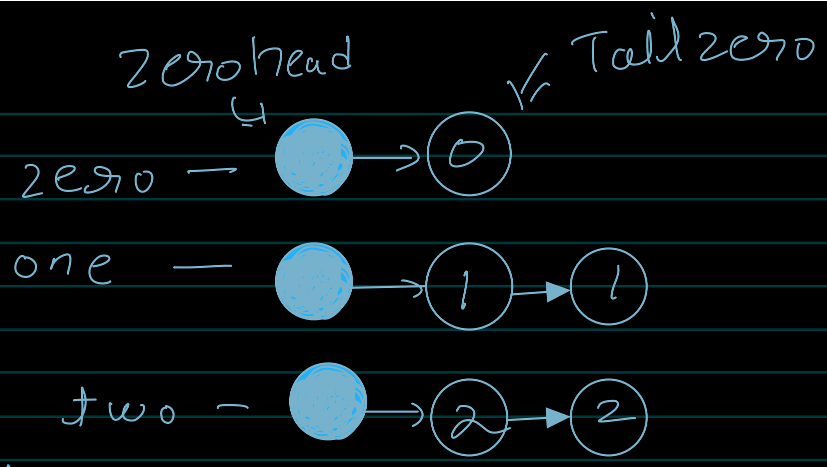
2 pointer pointing at some node

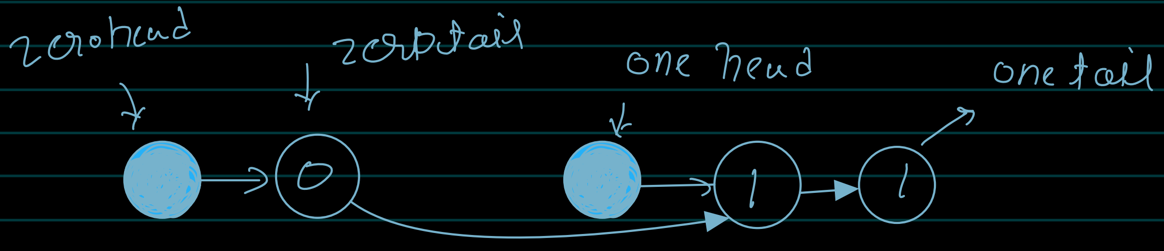


tail → next = (Node we got from unsorted LL)

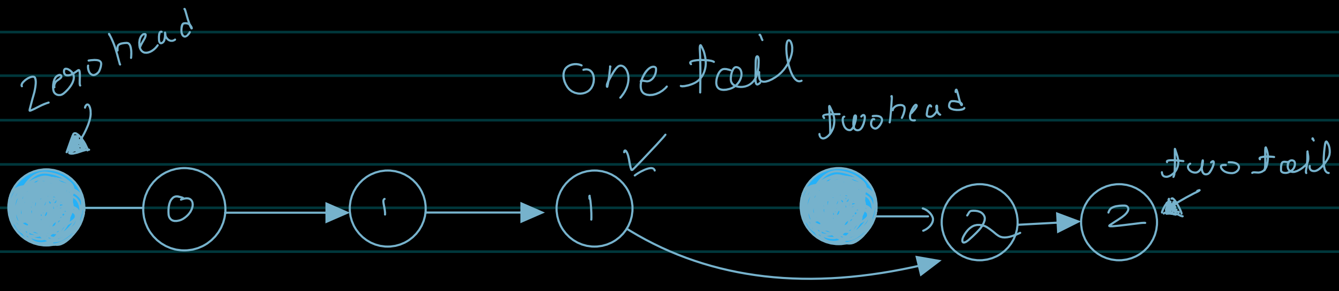
Shift tail to next ptr

tail = tail→next

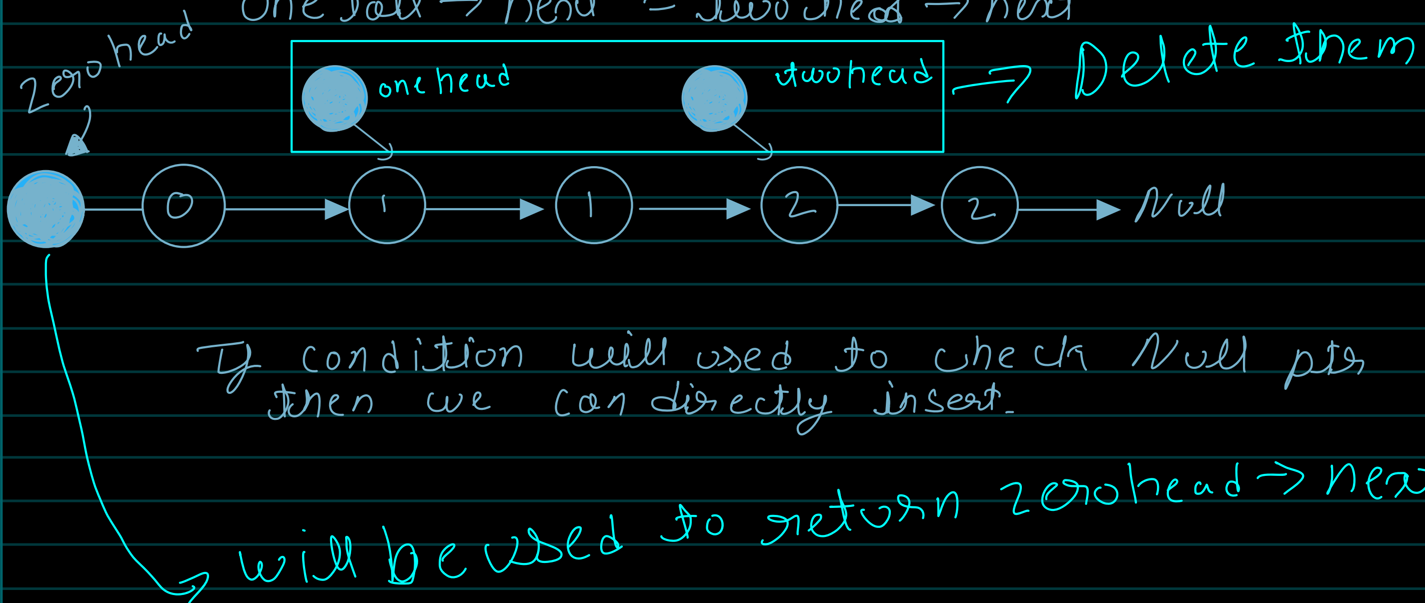




zero tail → next = one head → next



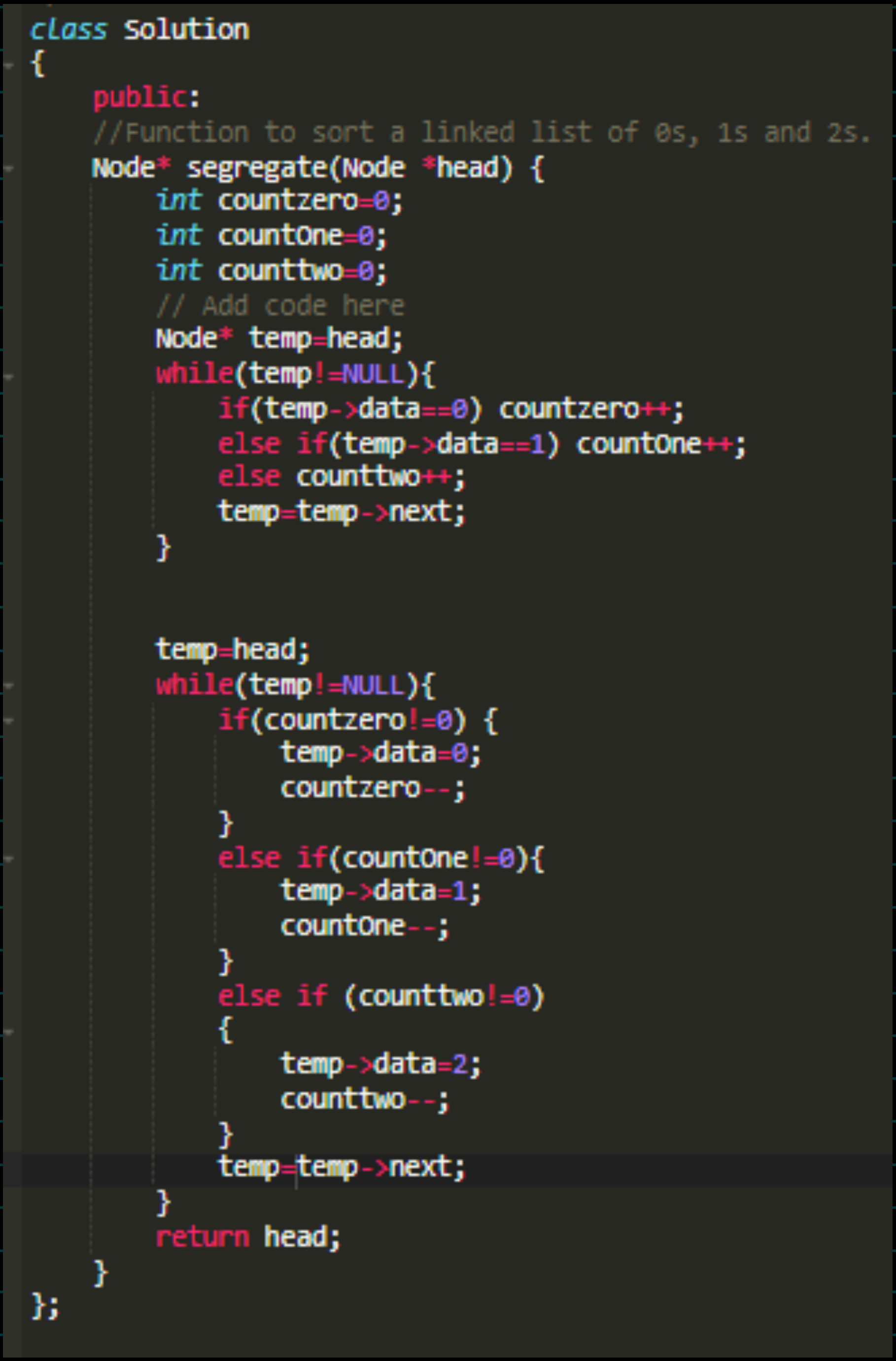
one tail → next = two head → next



If condition will used to check Null ptr then we can directly insert.

will be used to return zerohead → next

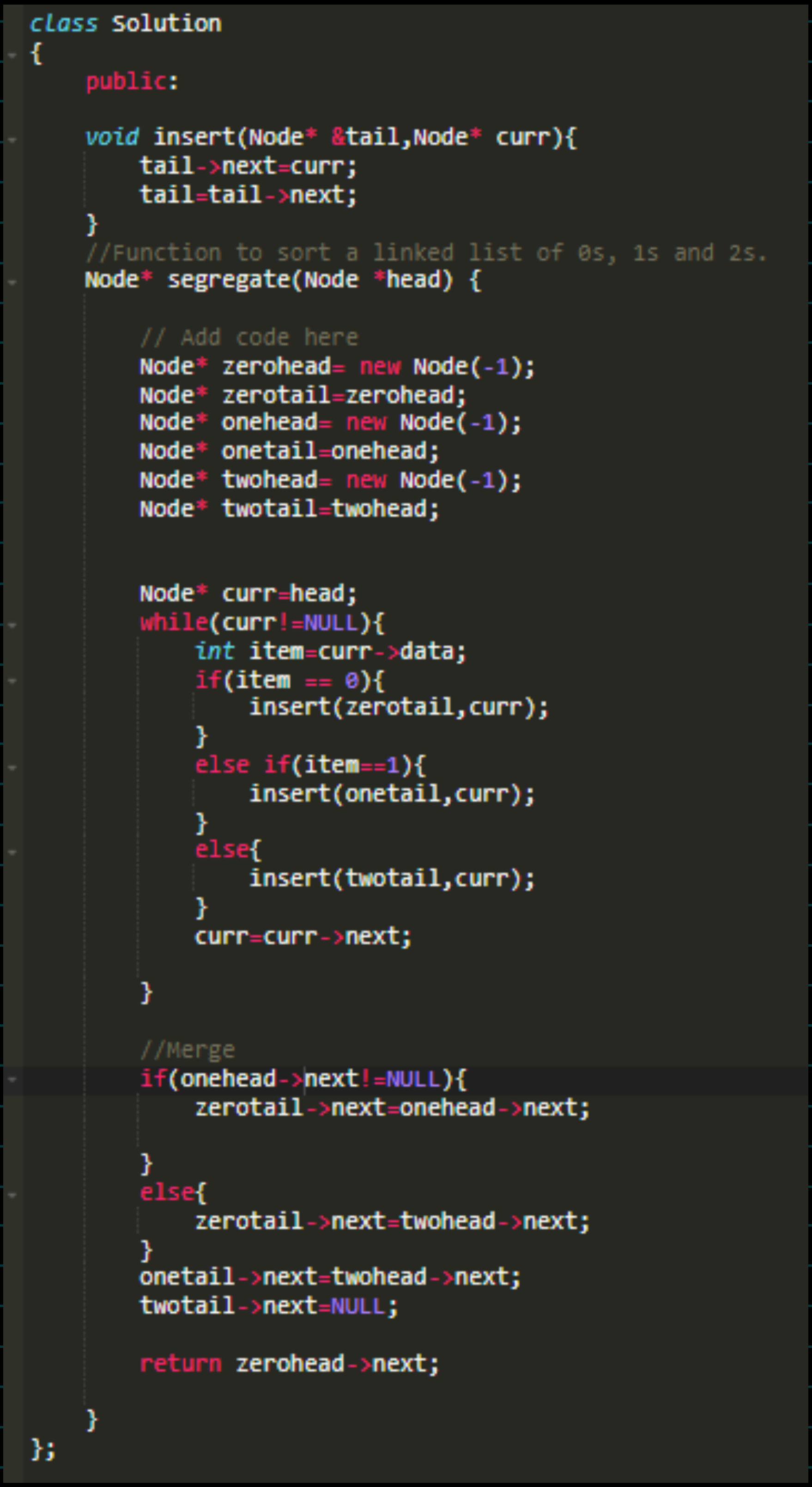
Approach:1 code



```
class Solution
{
    public:
    //Function to sort a linked list of 0s, 1s and 2s.
    Node* segregate(Node *head) {
        int countzero=0;
        int countOne=0;
        int counttwo=0;
        // Add code here
        Node* temp=head;
        while(temp!=NULL){
            if(temp->data==0) countzero++;
            else if(temp->data==1) countOne++;
            else counttwo++;
            temp=temp->next;
        }


        temp=head;
        while(temp!=NULL){
            if(countzero!=0) {
                temp->data=0;
                countzero--;
            }
            else if(countOne!=0){
                temp->data=1;
                countOne--;
            }
            else if (counttwo!=0)
            {
                temp->data=2;
                counttwo--;
            }
            temp=temp->next;
        }
        return head;
    }
};
```

Approach 2: code



```cpp
class Solution
{
    public:

    void insert(Node* &tail,Node* curr){
        tail->next=curr;
        tail=tail->next;
    }
    //Function to sort a linked list of 0s, 1s and 2s.
    Node* segregate(Node *head) {

        // Add code here
        Node* zerohead= new Node(-1);
        Node* zerotail=zerohead;
        Node* onehead= new Node(-1);
        Node* onetail=onehead;
        Node* twohead= new Node(-1);
        Node* twotail=twohead;


        Node* curr=head;
        while(curr!=NULL){
            int item=curr->data;
            if(item == 0){
                insert(zerotail,curr);
            }
            else if(item==1){
                insert(onetail,curr);
            }
            else{
                insert(twotail,curr);
            }
            curr=curr->next;

        }

        //Merge
        if(onehead->next!=NULL){
            zerotail->next=onehead->next;

        }
        else{
            zerotail->next=twohead->next;
        }
        onetail->next=twohead->next;
        twotail->next=NULL;

        return zerohead->next;

    }
};
```