Tuesday, January 18, 2022 3:31 PM

Given a linked list of 0s, 1s and 2s, sort it.
Easy Accuracy: 49.37% Submissions: 65801 Points: 2
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(1).

```
// { Driver Code Starts
//Initial Template for Java
import java.util.*;
import java.lang.*;
import java.io.*;
class Node
{
    int data;
    Node next;
    Node(int key)
```

```
    {
        data = key;
        next = null;
    }
}
class Driverclass
{
    public static void main (String[] args)
    {
        Scanner sc= new Scanner(System.in);
        int t = sc.nextInt();

        while(t-- > 0)
        {
            int n = sc.nextInt();
            Node head = new Node(sc.nextInt());
            Node tail = head;
            while(n-- > 1){
                tail.next = new Node(sc.nextInt());
                tail = tail.next;
            }

              head = new Solution().segregate(head);
             printList(head);
            System.out.println();
        }
    }

    public static void printList(Node head)
    {
        if(head == null)
           return;

        Node temp = head;
        while(temp != null)
        {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
    }


}


// } Driver Code Ends

//User function Template for Java
/*
class Node
{
    int data;
    Node next;
    Node(int data)
    {
        this.data = data;
        next = null;
    }
```

```
}
*/
class Solution
{
    //Function to sort a linked list of 0s, 1s and 2s.
//     static Node segregate(Node head)
//     {
//         if(head==null || head.next==null)

//         return head;

//         // add your code here
//         Node headZero=null;
//         Node headOne=null;
//         Node headTwo=null;
//         Node temp=head ;
//         Node tempZero=null;
//         Node tempOne=null;
//         Node tempTwo=null;


//         while(temp!=null)
//         {


//   if(temp.data==0)
// {
//   if(headZero==null)
//                 {
//                     headZero=temp;
//                     tempZero=headZero;
//                 }else{
//                     tempZero.next=temp;
//                     tempZero=temp;

//                 }}




//                 //1


//  else if(temp.data==1)
// {
//  if(headOne==null)
//                 {
//                     headOne=temp;
//                     tempOne=headOne;
//                 }else{
//                     tempOne.next=temp;
//                     tempOne=temp;

//                 } }

//                 //2
```

```
//  else if(temp.data==2)
// {
//  if(headTwo==null)
//                  {
//                      headTwo=temp;
//                      tempTwo=headTwo;
//                  }else{
//                      tempTwo.next=temp;
//                      tempTwo=temp;

//                  }
// }
//              temp=temp.next;

//         }

//      tempZero.next=null;
//       tempOne.next=null;
//       tempTwo.next=null;

//         tempZero.next=headOne;
//         tempOne.next=headTwo;
//         return headZero;

//     }
 static Node segregate(Node head)
    {
        if(head==null|| head.next==null)
        return head ;

      int zeroCount=0;
      int oneCount=0;
      int twoCount=0;

      Node temp=head;

      while(temp!=null)
      {
           if(temp.data==0)
           {zeroCount++;}
           else if(temp.data==1)
           {oneCount++;}
           else{ twoCount++;}
           temp=temp.next;

      }
    //   System.out.println("0: "+zeroCount);
    //   System.out.println("1: "+oneCount);
    //   System.out.println("2: "+twoCount);
      temp=head;



      while(zeroCount-->0)
```

```
        {
            temp.data=0;
            temp=temp.next;
        }
        while(oneCount-->0)
        {
            temp.data=1;
            temp=temp.next;
        }while(twoCount-->0)
        {
            temp.data=2;
            temp=temp.next;
        }
return head ;    }
}
```

```
// Java Program to sort a linked list 0s, 1s
// or 2s by changing links
public class Sort012 {
    // Sort a linked list of 0s, 1s and 2s
    // by changing pointers.
    public static Node sortList(Node head)
    {
        if(head==null || head.next==null)
        {
            return head;
        }
        // Create three dummy nodes to point to
        // beginning of three linked lists. These
        // dummy nodes are created to avoid many
        // null checks.
        Node zeroD = new Node(0);
        Node oneD = new Node(0);
        Node twoD = new Node(0);
        // Initialize current pointers for three
```

```
        // lists and whole list.
        Node zero = zeroD, one = oneD, two = twoD;
        // Traverse list
        Node curr = head;
        while (curr!=null)
        {
            if (curr.data == 0)
            {
                zero.next = curr;
                zero = zero.next;
                curr = curr.next;
            }
            else if (curr.data == 1)
            {
                one.next = curr;
                one = one.next;
                curr = curr.next;
            }
            else
            {
                two.next = curr;
                two = two.next;
                curr = curr.next;
            }
        }
        // Attach three lists
        zero.next = (oneD.next!=null)
? (oneD.next) : (twoD.next);
        one.next = twoD.next;
        two.next = null;
        // Updated head
        head = zeroD.next;
        return head;
    }
```