**Union of Two Linked Lists :-**

Easy Accuracy: 58.65% Submissions: 33K+ Points: 2

Given two linked lists, your task is to complete the function **makeUnion(),**that returns the union list of two linked lists. This union list should include all the **distinct**elements only and it should be sorted in **ascending**order.

**Example 1:**

**Input:**

L1 = 9->6->4->2->3->8

L2 = 1->2->8->6->2

**Output:**1 2 3 4 6 8 9  
**Explaination:**   
All the distinct numbers from two lists, when sorted forms the list in the output.

**Example 2:**

**Input:**

L1 = 1->5->1->2->2->5

L2 = 4->5->6->7->1

**Output:**1 2 4 5 6 7  
**Explaination:**   
All the distinct numbers from two lists, when sorted forms the list in the output.

**Your Task:**  
The task is to complete the function **makeUnion**() which makes the union of the given two lists and returns the head of the new list.

**Expected Time Complexity:** O((N+M)\*Log(N+M))  
**Expected Auxiliary Space:** O(N+M)

**Constraints:**  
1<=N,M<=104

**Code :-**

//{ Driver Code Starts

#include<bits/stdc++.h>

using namespace std;

struct Node

{

int data;

struct Node\* next;

Node(int x){

data = x;

next = NULL;

}

};

// } Driver Code Ends

class Solution

{

public:

struct Node\* makeUnion(struct Node\* head1, struct Node\* head2)

{

set<int> s;

Node \*ptr = head1;

while(ptr){

s.insert(ptr->data);

ptr = ptr->next;

}

ptr = head2;

while(ptr){

s.insert(ptr->data);

ptr = ptr->next;

}

Node \*anshead=NULL, \*prev=NULL, \*current;

for(auto item:s){

current = new Node(item);

if(!anshead)

anshead = current;

if(prev)

prev->next = current;

prev = current;

}

return anshead;

}

};

//{ Driver Code Starts.

struct Node\* buildList(int size)

{

int val;

cin>> val;

Node\* head = new Node(val);

Node\* tail = head;

for(int i=0; i<size-1; i++)

{

cin>> val;

tail->next = new Node(val);

tail = tail->next;

}

return head;

}

void printList(Node\* n)

{

while(n)

{

cout<< n->data << " ";

n = n->next;

}

cout<< endl;

}

int main()

{

int t;

cin>>t;

while(t--)

{

int n, m;

cin>>n;

Node\* first = buildList(n);

cin>>m;

Node\* second = buildList(m);

Solution obj;

Node\* head = obj.makeUnion(first,second);

printList(head);

}

return 0;

}

// } Driver Code Ends

**T.C :- O((N+M)\*log(N+M))**

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