*Note: Your solution should have O(l1.length + l2.length) time complexity, since this is what you will be asked to accomplish in an interview.*

Given two singly linked lists sorted in non-decreasing order, your task is to merge them. In other words, return a singly linked list, also sorted in non-decreasing order, that contains the elements from both original lists.

Example

* For l1 = [1, 2, 3] and l2 = [4, 5, 6], the output should be  
  mergeTwoLinkedLists(l1, l2) = [1, 2, 3, 4, 5, 6];
* For l1 = [1, 1, 2, 4] and l2 = [0, 3, 5], the output should be  
  mergeTwoLinkedLists(l1, l2) = [0, 1, 1, 2, 3, 4, 5].

Input/Output

* **[execution time limit] 3 seconds (cs)**
* **[input] linkedlist.integer l1**

A singly linked list of integers.

*Guaranteed constraints:*  
0 ≤ list size ≤ 104,  
-109 ≤ element value ≤ 109.

* **[input] linkedlist.integer l2**

A singly linked list of integers.

*Guaranteed constraints:*  
0 ≤ list size ≤ 104,  
-109 ≤ element value ≤ 109.

* **[output] linkedlist.integer**
  + A list that contains elements from both l1and l2, sorted in non-decreasing order.

<https://app.codesignal.com/interview-practice/task/6rE3maCQwrZS3Mm2H/description>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

class ListNode<T>

{

public T value { get; set; }

public ListNode<T> next { get; set; }

}

static ListNode<int> mergeTwoLinkedLists(ListNode<int> headA, ListNode<int> headB)

{

/\* a dummy first node to

hang the result on \*/

ListNode<int> dummyNode = new ListNode<int>(); // new Node(0);

dummyNode.value = 0;

/\* tail points to the

last result node \*/

ListNode<int> tail = dummyNode;

while (true)

{

/\* if either list runs out,

use the other list \*/

if (headA == null)

{

tail.next = headB;

break;

}

if (headB == null)

{

tail.next = headA;

break;

}

/\* Compare the data of the two

lists whichever lists' data is

smaller, append it into tail and

advance the head to the next Node

\*/

if (headA.value <= headB.value)

{

tail.next = headA;

headA = headA.next;

}

else

{

tail.next = headB;

headB = headB.next;

}

/\* Advance the tail \*/

tail = tail.next;

}

return dummyNode.next;

}

static void Main(string[] args)

{

Console.ReadLine();

}

}

}