// 21-Merge\_Two\_Sorted\_Lists.cpp : //

#include <iostream>

#include <list>

using namespace std;

/\*\*

\* Definition for singly-linked list.

\*/

struct ListNode {

int val;

ListNode \*next;

ListNode() : val(0), next(nullptr) {}

ListNode(int x) : val(x), next(nullptr) {}

ListNode(int x, ListNode \*next) : val(x), next(next) {}

};

class Solution {

public:

ListNode\* mergeTwoLists(ListNode\* l1, ListNode\* l2) {

//while (1)

//{

if (l1 == nullptr) {

return l2;

}

else if (l2 == nullptr) {

return l1;

}

else if (l1->val < l2->val) {

l1->next = mergeTwoLists(l1->next, l2);

return l1;

}

else {

l2->next = mergeTwoLists(l1, l2->next);

return l2;

}

//}

}

};

/\* Function to print nodes in a given linked list \*/

void printList(ListNode\* node)

{

while (node != NULL)

{

cout << node->val << "->";

node = node->next;

}

}

int main()

{

Solution sol;

#if 0

list<int> list1 = { 1, 2, 4 };

list<int> list2 = { 1, 3, 4 };

list2.merge(list1);

cout << "List: ";

for (auto it = list2.begin(); it != list2.end(); ++it)

cout << \*it << " ";

#else

/\* Start with the empty list \*/

ListNode\* res = NULL;

ListNode\* a = new ListNode(1);

ListNode\* b = new ListNode(1);

a->next = new ListNode(2);

a->next->next = new ListNode(4);

b->next = new ListNode(3);

b->next->next = new ListNode(4);

res = sol.mergeTwoLists(a, b);

printList(res);

#endif

return 0;

}