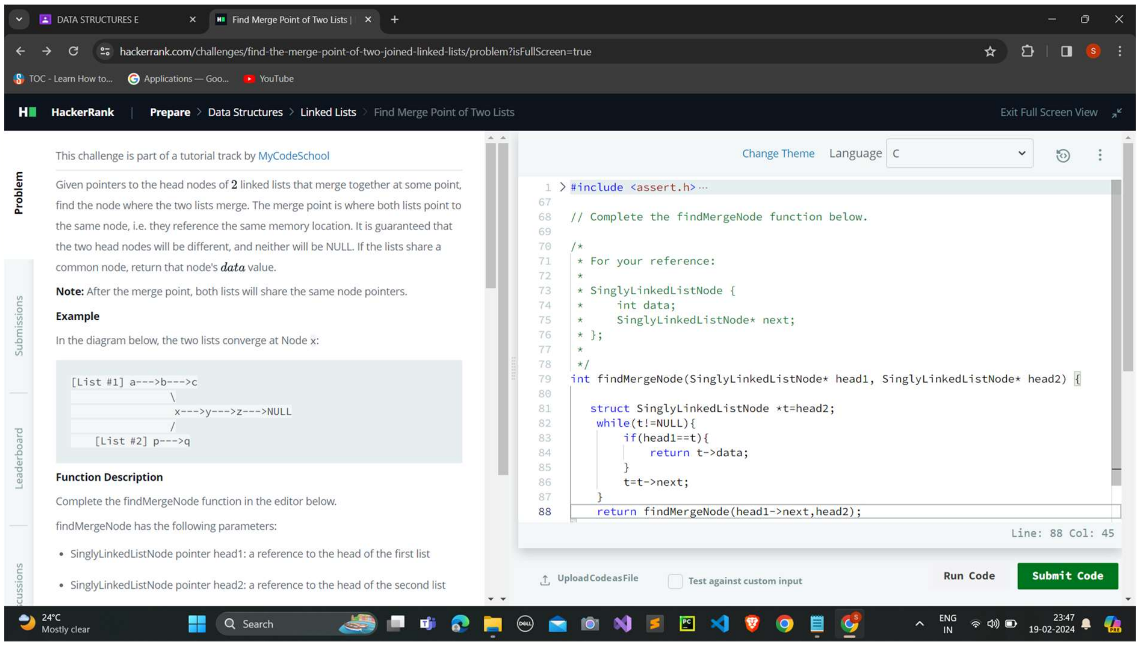
HACKERRANK QUESTION: Find Merge Point of Two Lists

```
//NAME: SHIVARAJ K PUJARI
//USN: 1BM22CS259
#include <assert.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();

typedef struct SinglyLinkedListNode SinglyLinkedListNode;
typedef struct SinglyLinkedList SinglyLinkedList;

struct SinglyLinkedListNode {
    int data;
    SinglyLinkedListNode* next;
};

struct SinglyLinkedList {
    SinglyLinkedListNode* head;
    SinglyLinkedListNode* tail;
};

SinglyLinkedListNode* create_singly_linked_list_node(int node_dat
a) {
    SinglyLinkedListNode* node = malloc(sizeof(SinglyLinkedListNo
de));

    node->data = node_data;
    node->next = NULL;

    return node;
}

void insert_node_into_singly_linked_list(SinglyLinkedList** singl
y_linked_list, int node_data) {
    SinglyLinkedListNode* node = create_singly_linked_list_node(n
ode_data);

    if (!(*singly_linked_list)->head) {
        (*singly_linked_list)->head = node;
    } else {
        (*singly_linked_list)->tail->next = node;
    }

    (*singly_linked_list)->tail = node;
}

void print_singly_linked_list(SinglyLinkedListNode* node, char* s
ep, FILE* fptr) {
    while (node) {
        fprintf(fptr, "%d", node->data);

        node = node->next;

        if (node) {
            fprintf(fptr, "%s", sep);
        }
    }
}

void free_singly_linked_list(SinglyLinkedListNode* node) {
    while (node) {
        SinglyLinkedListNode* temp = node;
        node = node->next;

        free(temp);
    }
}

// Complete the findMergeNode function below.

/*
 * For your reference:
 *
 * SinglyLinkedListNode {
 *     int data;
 *     SinglyLinkedListNode* next;
 * };
 *
 */
int findMergeNode(SinglyLinkedListNode* head1, SinglyLinkedListNo
de* head2) {

   struct SinglyLinkedListNode *t=head2;
    while(t!=NULL){
        if(head1==t){
            return t->data;
        }
        t=t->next;
    }
    return findMergeNode(head1->next,head2);
}

int main()
{
    FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

    char* tests_endptr;
    char* tests_str = readline();
    int tests = strtol(tests_str, &tests_endptr, 10);

    if (tests_endptr == tests_str || *tests_endptr != '\0') { exi
t(EXIT_FAILURE); }

    for (int tests_itr = 0; tests_itr < tests; tests_itr++) {
        char* index_endptr;
        char* index_str = readline();
        int index = strtol(index_str, &index_endptr, 10);

        if (index_endptr == index_str || *index_endptr != '\0') {
 exit(EXIT_FAILURE); }

        SinglyLinkedList* llist1 = malloc(sizeof(SinglyLinkedList
));
        llist1->head = NULL;
        llist1->tail = NULL;

        char* llist1_count_endptr;
        char* llist1_count_str = readline();
        int llist1_count = strtol(llist1_count_str, &llist1_count
_endptr, 10);

        if (llist1_count_endptr == llist1_count_str || *llist1_co
unt_endptr != '\0') { exit(EXIT_FAILURE); }

        for (int i = 0; i < llist1_count; i++) {
            char* llist1_item_endptr;
            char* llist1_item_str = readline();
            int llist1_item = strtol(llist1_item_str, &llist1_ite
m_endptr, 10);
```

```
            if (llist1_item_endptr == llist1_item_str || *llist1_
item_endptr != '\0') { exit(EXIT_FAILURE); }

            insert_node_into_singly_linked_list(&llist1, llist1_i
tem);
        }

        SinglyLinkedList* llist2 = malloc(sizeof(SinglyLinkedList
));
        llist2->head = NULL;
        llist2->tail = NULL;

        char* llist2_count_endptr;
        char* llist2_count_str = readline();
        int llist2_count = strtol(llist2_count_str, &llist2_count
_endptr, 10);

        if (llist2_count_endptr == llist2_count_str || *llist2_co
unt_endptr != '\0') { exit(EXIT_FAILURE); }

        for (int i = 0; i < llist2_count; i++) {
            char* llist2_item_endptr;
            char* llist2_item_str = readline();
            int llist2_item = strtol(llist2_item_str, &llist2_ite
m_endptr, 10);

            if (llist2_item_endptr == llist2_item_str || *llist2_
item_endptr != '\0') { exit(EXIT_FAILURE); }

            insert_node_into_singly_linked_list(&llist2, llist2_i
tem);
        }

        SinglyLinkedListNode* ptr1 = llist1->head;
        SinglyLinkedListNode* ptr2 = llist2->head;

        for (int i = 0; i < llist1_count; i++) {
            if (i < index) {
                ptr1 = ptr1->next;
            }
        }

        for (int i = 0; i < llist2_count; i++) {
            if (i != llist2_count-1) {
                ptr2 = ptr2->next;
            }
        }

        ptr2->next = ptr1;

        int result = findMergeNode(llist1->head, llist2->head);

        fprintf(fptr, "%d\n", result);
    }

    fclose(fptr);

    return 0;
}

char* readline() {
    size_t alloc_length = 1024;
    size_t data_length = 0;
    char* data = malloc(alloc_length);

    while (true) {
        char* cursor = data + data_length;
        char* line = fgets(cursor, alloc_length - data_length, st
din);

        if (!line) { break; }

        data_length += strlen(cursor);

        if (data_length < alloc_length - 1 || data[data_length -
1] == '\n') { break; }

        size_t new_length = alloc_length << 1;
        data = realloc(data, new_length);

        if (!data) { break; }

        alloc_length = new_length;
    }

    if (data[data_length - 1] == '\n') {
        data[data_length - 1] = '\0';
    }

    data = realloc(data, data_length);
    return data;
}
```



DATA STRUCTURES E | Find Merge Point of Two Lists |

hackerrank.com/challenges/find-the-merge-point-of-two-joined-linked-lists/problem?isFullScreen=true

TOC - Learn How to... | Applications — Goo... | YouTube

HackerRank | Prepare > Data Structures > Linked Lists > Find Merge Point of Two Lists | Exit Full Screen View

This challenge is part of a tutorial track by MyCodeSchool

Given pointers to the head nodes of 2 linked lists that merge together at some point, find the node where the two lists merge. The merge point is where both lists point to the same node, i.e. they reference the same memory location. It is guaranteed that the two head nodes will be different, and neither will be NULL. If the lists share a common node, return that node's *data* value.

Note: After the merge point, both lists will share the same node pointers.

Example

In the diagram below, the two lists converge at Node x:

```
[List #1] a--->b--->c
                     \
                      x--->y--->z--->NULL
                     /
    [List #2] p--->q
```

Function Description

Complete the findMergeNode function in the editor below.

findMergeNode has the following parameters:

- SinglyLinkedListNode pointer head1: a reference to the head of the first list
- SinglyLinkedListNode pointer head2: a reference to the head of the second list

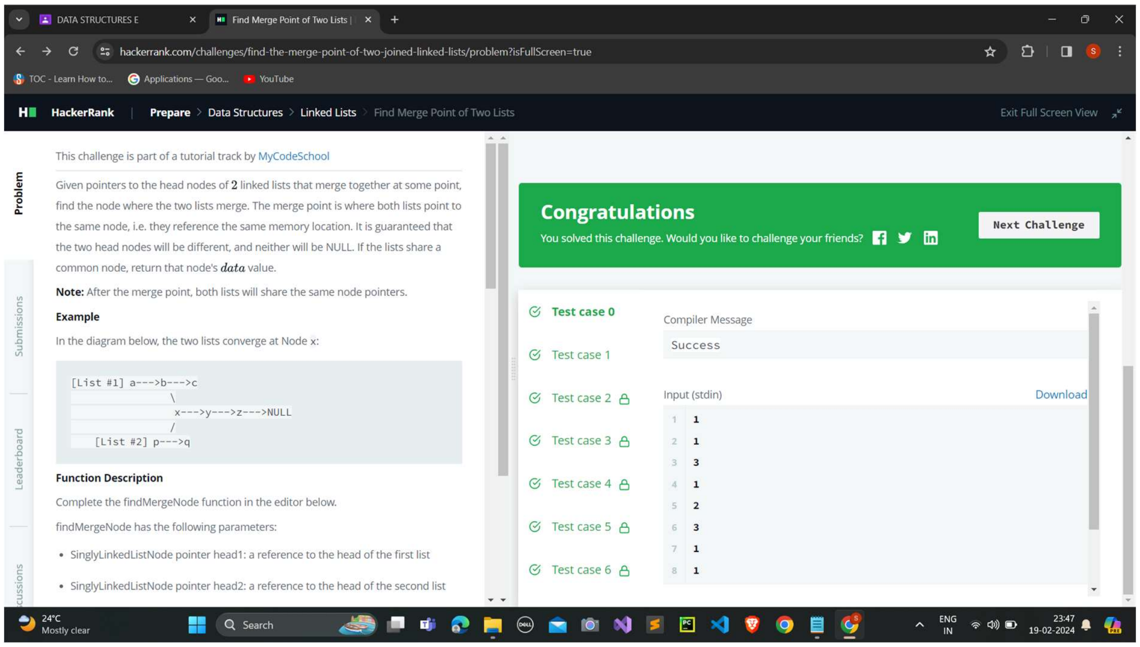
Change Theme | Language C

```c
> #include <assert.h> ···

// Complete the findMergeNode function below.

/*
 * For your reference:
 *
 * SinglyLinkedListNode {
 *     int data;
 *     SinglyLinkedListNode* next;
 * };
 *
 */
int findMergeNode(SinglyLinkedListNode* head1, SinglyLinkedListNode* head2) {

   struct SinglyLinkedListNode *t=head2;
    while(t!=NULL){
        if(head1==t){
            return t->data;
        }
        t=t->next;
    }
    return findMergeNode(head1->next,head2);
```

Line: 88 Col: 45

Upload Code as File | Test against custom input | Run Code | Submit Code

24°C Mostly clear | Search | ENG IN | 23:47 19-02-2024



DATA STRUCTURES E | Find Merge Point of Two Lists |

hackerrank.com/challenges/find-the-merge-point-of-two-joined-linked-lists/problem?isFullScreen=true

HackerRank | Prepare > Data Structures > Linked Lists > Find Merge Point of Two Lists | Exit Full Screen View

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0 | Test case 1 | Test case 2 | Test case 3 | Test case 4 | Test case 5 | Test case 6

Compiler Message

```
Success
```

Input (stdin) Download

```
1
1
3
1
2
3
1
1
```

24°C Mostly clear | Search | ENG IN | 23:47 19-02-2024