Reverse a linked list

locked

Problem | Submissions | Leaderboard | Discussions

This challenge is part of a tutorial track by MyCodeSchool and is accompanied by a video lesson.

You're given the pointer to the head node of a linked list. Change the `next` pointers of the nodes so that their order is reversed. The head pointer given may be null meaning that the initial list is empty.

Submissions: 131
Max Score: 25
Difficulty: Easy

Rate This Challenge:

More

Input Format

You have to complete the `SinglyLinkedListNode reverse(SinglyLinkedListNode head)` method which takes one argument - the head of the linked list. You should NOT read any input from stdin/console.

The input is handled by the code in the editor and the format is as follows:

The first line contains an integer t, denoting the number of test cases.
Each test case is of the following format:

The first line contains an integer n, denoting the number of elements in the linked list.
The next n lines contain an integer each, denoting the elements of the linked list.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n \leq 1000$
- $1 \leq list_i \leq 1000$, where $list_i$ is the i^{th} element in the list.

Output Format

Change the `next` pointers of the nodes that their order is reversed and `return` the head of the reversed linked list. Do NOT print anything to stdout/console.

The output is handled by the code in the editor. The output format is as follows:

For each test case, print in a new line the elements of the linked list after reversing it, separated by spaces.

Sample Input

```
1
5
1
2
3
4
5
```

Sample Output

```
5 4 3 2 1
```

Explanation

The initial linked list is: 1 -> 2 -> 3 -> 4 -> 5 -> NULL

The reversed linked list is: 5 -> 4 -> 3 -> 2 -> 1 -> NULL

Current Buffer (saved locally, editable) C++

```
#include ⇔

using namespace std;

class SinglyLinkedListNode {
    public:
        int data;
        SinglyLinkedListNode *next;

        SinglyLinkedListNode(int node_data) {
            this->data = node_data;
            this->next = nullptr;
        }
};

class SinglyLinkedList {
    public:
        SinglyLinkedListNode *head;
        SinglyLinkedListNode *tail;

        SinglyLinkedList() {
            this->head = nullptr;
            this->tail = nullptr;
        }

        void insert_node(int node_data) {
            SinglyLinkedListNode* node = new SinglyLinkedListNode(node_data);

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

            this->tail = node;
        }
};

void print_singly_linked_list(SinglyLinkedListNode* node, string sep, ofstream& fout) {
    while (node) {
        fout << node->data;

        node = node->next;

        if (node) {
            fout << sep;
        }
    }
}

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

        free(temp);
    }
}

// Complete the reverse function below.

/*
 * For your reference:
 *
 * SinglyLinkedListNode {
 *     int data;
 *     SinglyLinkedListNode* next;
 * };
 *
 */
SinglyLinkedListNode* reverse(SinglyLinkedListNode* head) {
SinglyLinkedListNode* prev=nullptr;
SinglyLinkedListNode* cur=nullptr;
SinglyLinkedListNode* next=nullptr;
cur=head;
while(cur!=nullptr){
    next=cur->next;
    cur->next=prev;
    prev=cur;
    cur=next;;
}
head=prev;
return head;
}


int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    int tests;
    cin >> tests;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    for (int tests_itr = 0; tests_itr < tests; tests_itr++) {
        SinglyLinkedList* llist = new SinglyLinkedList();

        int llist_count;
        cin >> llist_count;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        for (int i = 0; i < llist_count; i++) {
            int llist_item;
            cin >> llist_item;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            llist->insert_node(llist_item);
        }

        SinglyLinkedListNode* llist1 = reverse(llist->head);

        print_singly_linked_list(llist1, " ", fout);
        fout << "\n";

        free_singly_linked_list(llist1);
    }

    fout.close();

    return 0;
}
```

Line: 29 Col: 1

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