ECE 26400 Practica 2-c

1 Resources

Resources you **can** use:

- The C manual pages: in the terminal, type `man [function]` to open the manpages for the given function.
- Printed notes, as many as you want! Scratch paper and a pencil may also be useful.

You **cannot** use anything not mentioned above, including digital notes and online resources.

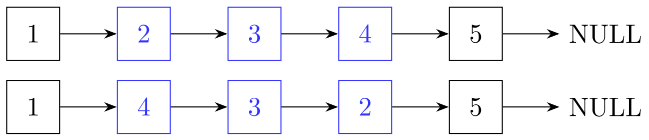
2 Problem: Reverse Range of Linked List

Given the `head` of a linked list and two 0-indexed integers `left` and `right` where `left < right`, write a program that reverses the nodes of the sublist defined on the **index** (not value) interval `[left, right)`.

Write your solution in the `reverse_range` function in `practica_c.c`. You may add helper functions if you would like. The `reverse_range` function has three parameters, the list head (`ListNode *head`), the left position, and the right position integers. Your solution should **not** create a new linked list (no malloc).

The `ListNode` struct is defined in `practica_c.h`. Ensure you are familiar with its members before writing code.

Example 1: (See next page for full figure and more examples)



The above figure shows the case for `head = [1, 2, 3, 4, 5]`, `left = 1`, `right = 4`. The sublist `[2, 3, 4]` is reversed to `[4, 3, 2]`, and the left-end nodes `[1]` and `[4]` are updated, resulting in output `[1, 4, 3, 2, 5]`.

Example 2:

Consider `head = [3, 4, 1]`, `left = 0`, `right = 1`. `3` is the only element in indexes $[0, 1)$, so return `[3, 4, 1]`.

2.1 Constraints

- The number of nodes in the given list will range from `[1, 100000]`.
- The number of queries (# of times run consecutively on the same list) will range from `[1, 1000]`.
- If either bound `left` or `right` is out of bounds, do **nothing** (return the list as is)
- The `const int idx` member of struct `ListNode` is **constant** (You cannot change it).

2.2 Testing with Make

Run the below commands in your terminal to compile and test your code.

- `make test`*x* - Run test x where x is a number 1-10. Running `make testall` will run all tests.
- `make leak` - Run valgrind for memory leak checking.

3 Submission

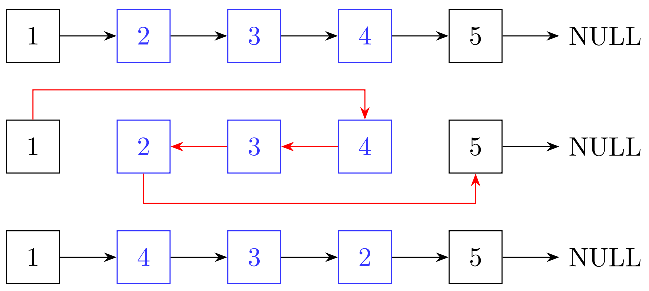
Submit only `practica_c.c` on Gradescope to run the autograder and get your score. Once you are finished, raise your hand and wait for a TA. They will check you off, after which you are allowed to leave.

4 Hints

- Observe the example diagrams, paying special attention to how the next pointers are change from step to step.
- What nodes need to be kept track of in order to correctly incorporate the reversed sublist to the surrounding, unchanged list?

5 Examples

Ex 1. (expanded):



Input: `head = [1, 2, 3, 4, 5]`, `left = 1`, `right = 4`

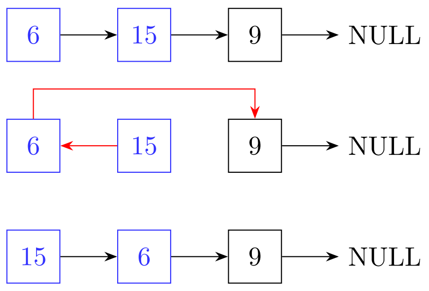
Output: `[1, 4, 3, 2, 5]`

Explanation: The given index range is `[1, 4)`, which is the sublist `[2, 3, 4]`. Reversing this sublist results in `[4, 3, 2]`. The node previous to the range start `Node 1` is updated to point to the new head of the reversed list, and the tail of the reversed list `Node 2` is updated to point to the node after the range end, giving the expected output: `[1, 4, 3, 2, 5]`.

Ex 2.

See first page, which demonstrates an edge case.

Ex 3.



Input: `head = [6, 15, 9]`, `left = 0`, `right = 2`

Output: `[15, 6, 9]`

Explanation: The given index range is `[0, 2)`, which is the sublist `[6, 15]`. Reversing the sublist gives `[15, 6]`. Since `Node 6` is the head node, we do not need to update the previous node. First, we update the head (`Node 6`) to point to the node after the range (`Node 9`). Then, we update the head to become the head of the reversed list (from `head = Node 6` to `head = Node 15`). This results in the expected output: `[15, 6, 9]`.