IITM-CS2710: Programming & Data Structures Lab (Jul-Nov 2018) Lab Session 7

Sep 17, 2018 **List ADT using Doubly Linked Lists** Mode: On Hacker Rank

Overview:

This assignment has only take-home component since there is no lab session this week. The contest is set up on hacker rank at the link https://www.hackerrank.com/iitm-pds-lab7 The submission deadline is set to `- Sat Sep 22 23:59:00 IST 2018 -` on hacker-rank link.

In this assignment you will implement the List ADT using doubly linked list. We have done this in the theory lecture class a few days before, and we will have some additional member functions modifying the list now. Unlike in previous assigments, you **will not** have to read input. Code for input-output operations is already written in HackerRank. You will only have to fill in the member functions given. And more importantly, edits are disabled for the main function, by the hacker rank settings for this contest.

Detailed Structure of Assignment task:

The design that you need to do is as follows :

`ListNode` **struct:**

- An instance of this struct represents a single node in the list. `val` is the `int` data the node contains, `next` points to the next node and `prev` points to the previous node.

`List` **class:**

- An object of this class represents a list, the class has two private members `begin`, `end` which point to the first and last node of the list respectively. When the list is empty, both `begin`, `end` are `NULL`.

Functions you will have to implement in the `List` class:

- Note: In the below text, `<node i>` is a pointer to the node with `val = i`
- `ListNode* push_back(int val)`: Creates a new `ListNode` with data `val` and inserts the node at the end of the list. Returns a pointer to the created node.
- `ListNode* push_front(int val)`: Creates a new `ListNode` with data `val` and inserts the node at the beginning of the list. Returns a pointer to the created node.
- `ListNode* insert(ListNode *ln, int val)`: Create a new node with data `val` and insert the node before the given node. Return a pointer to the inserted node. E.g. if the list is `1->2->3->4` and `insert(<node 2>, 6)` is called, the list becomes `1->6->2->3->4` and `<node 6>` is returned.
- `bool is_empty()`: Checks if the list is empty, returns `true` if it is, `false` otherwise.

- `void move_to_front(ListNode *ln)`: Detaches the given node from the list, and inserts it at the beginning. Note that you should not create a new node for this.
- `void swap_pairs()`: Swap nodes in adjacent node-pairs in the list. E.g: If the list is `1->2->3->4`, after calling `swap_pairs` it should become `2->1->4->3`. If the list is `1->2->3`, then it becomes `2->1->3`. Note that you should not simply swap the values of adjacent nodes, you will have to change the pointers such the the entire nodes are swapped. *The output checks are such that you will not match the test case outputs if you simply swap the values in the function that you write.*
- `void reverse_segment(ListNode *ln1, ListNode *ln2)`: Reverse the list from the nodes `ln1` to `ln2`. E.g: if the list is `1->2->3->4`, and `reverse_segment(<node 1>, <node 3>)` is called, then the list becomes `3->2->1->4`
- `void splice(ListNode *ln, List l)`: Merge the list `l` into the current list before the node `ln`. E.g: If the list is `1->2->3->4` and `l` is `10->11->12`, calling `splice(<node 3>, l)` should make current list `1->2->10->11->12->3->4`. Note that you should not create new nodes for the operation.
- `void print()`: Print node values in the list to `cout`. E.g: if list is `1->2->3->4` , print `1 2 3 4` to `cout`.
- See https://ideone.com/1NGxWl for the template.