Data Structures

SLL Homework 1

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Tip

- Through the whole course homework you must:
  - Compute time order
  - Compute memory order
  - Skip the effect of the debug_data items. They are for educational purposes

Problem #1: Destructor

- Our code used to create dynamic data, but never released
- This creates a memory leak
- Implement destructor: ~LinkedList()
- It should free all the created nodes

Problem #2: Insert front

- We implemented insert_end in the lecture
- We want to be able to insert front as following

```
LinkedList list;

list.insert_end(6);
list.insert_end(10);
list.insert_end(8);
list.insert_end(15);

list.insert_front(7);
list.insert_front(5);
list.insert_front(1);

list.print();
// 1 5 7 6 10 8 15
```

Problem #3: Delete front

- The opposite of insert front

```
LinkedList list;

list.insert_end(6);
list.insert_end(10);
list.insert_end(8);
list.insert_end(15);

list.delete_front();
list.print();
// 10 8 15
```

Problem #4: Get nth from back

- We already implemented Node* get_nth(int n)
- Now implement: Node* get_nth_back(int n)
- Given 1-based position, find it from the back
- E..g. if list is 1 2 3 4 5 6
- get_nth_back(1) should point to node with value 6

Problem #5: Is Same list's data?

- Develop function to check if lists are data-equal:
  - Same length - each node and its corresponding one has same value
  - bool is_same(const LinkedList &other)
- Provide 2 codes
  - One code assumes a variable length is maintained
    That tells us how many nodes so far
    - E.g. in each insert, length is increased
  - Another that doesn't use it and don't compute length

```
LinkedList list1;
LinkedList list2;

cout<<list1.is_same(list2)<<"\n";    // 1
list1.insert_end(6);
list1.insert_end(10);
list2.insert_end(6);
cout<<list1.is_same(list2)<<"\n";    // 0
list2.insert_end(10);
cout<<list1.is_same(list2)<<"\n";    // 1
list1.insert_end(8);
list1.insert_end(15);
list2.insert_end(8);
list2.insert_end(77);
cout<<list1.is_same(list2)<<"\n";    // 0
```

Problem #6: Linked List without tail/length!

- Assume we will implement our linked list to only have head but no tail
- Implement and test these 3 methods
  - Add element just add element to our current collection of numbers
    - O(1)
    - Tip: data will be reversed
  - Get tail get the last node

```
class LinkedList {
private:
    Node *head { };
public:

    void print() {

    void add_element(int value) {

    Node* get_tail() {
};
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."