CS2710 - Programming and Data Structures Lab

Lab 5 (ungraded)

Aug 28, 2024

Instructions

- You are expected to solve ALL the problems in the lab, using the local computer, C++ language, and g++ compiler. (For graded labs, Bonus problems can fetch 5% capped bonus; and Optional problems are just for fun and won't be graded - hence, they need not be submitted and can be done outside the lab too.)
- You should submit your code to the course **moodle** on time, i.e., on/before 4.45pm (so that TAs can subsequently grade your submissions for graded lab assignments using the private test cases).
- You must strictly adhere to the following naming convention for your .cpp files and the single .zip file submission. For example, for a Lab Session 5 consisting of 2 questions, a student with roll number CS23B000 should
  - Name their .cpp files as **CS23B000_LAB5_Q1.cpp** and **CS23B000_LAB5_Q2.cpp**
  - Put both these .cpp files in a directory named **CS23B000_LAB5**
  - Zip this directory into a file named **CS23B000_LAB5.zip**
  - Submit only this single .zip file to moodle.
- For graded labs, if you need assistance, ask your TA, not your classmate.
- Internet access and mobile devices are prohibited in the lab.
- For graded labs, check course Moodle for the public test cases and the evaluation script, which you can use to test your programs.
- Solve the problems using array/vector/strings only (do not use any other data structure). You can implement other data structures you may need, and use std::sort as needed.

---

1. [LOGISTICS] Please read email sent to moodle on what today's lab session is about. Besides the practice/optional questions below, please clarify any doubts from earlier labs that you may've with TAs or friends, and you can also practice questions from earlier labs today. Today is also the deadline to raise regrade requests, if any, you have on earlier labs, with TAs.
2. [FROM HERE TO THERE AND BACK AGAIN!] Write recursive and iterative versions of **reverse**, which reverses a singly linked list. Do not create new Node objects; instead reuse the existing ones.

   You can start with the singly linked list code seen in class, and then add **reverse** as a new member function to the List class. As discussed in class, you may also want to try changing the template implementation in the code from
   "`template <typename Node> class List { private: Node *head; .... }`" to
   "`template <typename Element> class List { private: Node<Element> *head; .... }`".

3. [LOOK BOTH WAYS] Write a class to implement doubly linked list. You can start with the singly linked list code and revise all member functions to properly handle both prev and next pointers in each Node object.

4. [OPTIONAL: LOOP INVARIANTS AND LARGE LANGUAGE MODELS (LLMS)] Attempt this question after attempting all the above questions on your own. For this question, you can use your mobile phone and AI assistance. Use at least two LLMs (e..g, ChatGPT 4o mini LLM, or WhatsApp/Meta AI's Llama 3.1 LLM, or github co-pilot) to generate code for a singly or doubly linked list.

   (a) Do the codes generated by these LLMs look similar? How similar are these codes to the code you wrote for implementing a singly or doubled linked list class?

   (b) What are the loop invariants for the code generated by LLM for removing all occurrences of an element from a singly (or doubly) linked list? Using loop invariants, can you argue that the generated code is correct? Did you find any LLM-generated code to be buggy?

   (c) A programmer's overall coding time is the sum of development time and correctness testing/verification time - is the latter same whether verifying your own code or someone else's (e.g., LLM-generated) code?

   (d) Moving on from standard Linked Lists, let's put LLMs to test for implementing advanced linked lists such as probabilistic skip lists. You can read about skip lists in Wikipedia. Ask LLMs to generate recursive code to reverse a skip list. Is the code generated by it satisfactory to you? Did you have to do some "prompt engineering" to get the code logic/style you want?