Lab 03: Dynamic Memory and Linked Lists

Overview

This lab will practice the key concept of dynamic memory management as well as part of the implementation of a linked list.

Dynamic Memory Management

Allocating, deallocating, accessing, and modifying dynamic-allocated data in C++ can be challenging. If one does not fully understand how memory management works, non-optimized solutions will happen (in the best case scenario), and time-consuming bugs are likely to appear.

For all of the data structures seen in this course, we make use of dynamic memory allocated on the heap. Therefore, it is important that we discuss challenging scenarios that improve the understanding of this widely used concept.

For this lab, we focus on the challenges of memory allocation within different scopes, multi-level pointers (pointers to pointers), and pointer arithmetic. It can be helpful to draw a memory map of the different dynamically allocated structures you encounter in the examples.

Removing an Element from a Singly Linked List

In lecture, you saw examples of inserting elements at any index in a singly linked list. In this lab, we'll explore the implementation of removing an element from such a list. The process follows the pseudocode described in class.

You'll be implementing a function `deleteNode()`, which accepts a pointer to the start of the list and an index of what element to delete.

There are some key differences between this function and the one you are writing for the homework. First, this function is not embedded in a LinkedList class. Thus, we are passing a Node** for the head so that it exists on the heap, but in other examples you may have seen, the head pointer has already been on the heap, just inside a LinkedList object. Second, you do not need to return the deleted element, only free the memory. Third, recall that this is singly-linked, so you'll have to rely exclusively on the next node pointer to locate the index to delete and update the list upon deletion.

Hints: check for an empty list first, then handle the case of removing and updating the head when the position is 0. After that, you can tackle the problem of traversing and removing elsewhere in the list.

Submission

Complete the "Lab 03. Dynamic Memory and Linked Lists" activity on Mimir to acknowledge the completion of this lab work.

You will answer four questions on Mimir. The first three questions focus on dynamic memory management. The last question focuses on linked list deletion. A battery of automated tests will be run to ensure that the errors have been fixed and the results can be properly computed.

A grade of "complete" on this lab work requires a score of 85%.