Lab 7 - Linked Lists

CS 251, Fall 2024

Copyright Notice

Exam 2

If you are taking the exam in your lab section, **take the exam on a lab computer, not your personal laptop**. You can sign into a lab computer with your NetID and password.

Exams are individual, closed-book, and closed-internet. You may not use ChatGPT, look at others' screens, or communicate with others during the exam time. Academic misconduct on an exam will result in a failing grade in the course.

Access the exam through Blackboard. In the sidebar, there is a section labeled "Exams". You should see Mastery Exam 2 in this section.

Complete the exam before working on the lab worksheet. If you finish early, please be quiet and respectful of your peers who are still taking the exam.

Learning Goals

By the end of this lab, you'll:

- Understand how assigning and reassigning pointers changes the memory model
- Have practice with various ways of manipulating linked lists through manual assignment, iteration, and recursion

Starter Code

N/A

Tasks

For this lab worksheet, work on the tasks in the **given order**. We strongly encourage collaboration in labs. Work with your peers (once they're done with the exam), and ask your TAs questions!

Throughout, we will use the following `Node` definition:

```
struct Node {
  int data;
  Node* next;

  Node(int data) {
    this->data = data;
    this->next = nullptr;
  }
};
```

Trace through the following lines of code carefully, and **draw a diagram** to determine what memory looks like after each line. Ask TAs if you need scratch paper. To draw a diagram, follow these steps:

1. Evaluate the right hand side as a **value**.
2. Evaluate the left hand side pointer by either creating a new variable, or "looking up" an existing variable.
3. Assign (draw an arrow from) the LHS to the RHS.

Code Snippet 1

```
1  Node* first = new Node(10);
2  Node* temp = first->next;
```

Question 1:

- How many integer **values** were created in lines 1 and 2?
- How many `Node` **values** were created in lines 1 and 2?

Code Snippet 2

Assuming that all the previous lines have been executed, trace the following lines of code carefully to determine what memory looks like after each line.

```
3  first->next = new Node(100);
4  first->next->next = temp;
```

Question 2:

- How many `Node` **values** were created in lines 3 and 4?
- After line 4, there are three possible `Node` values for any `Node` pointer:
  - The `Node` with data 10
  - The `Node` with data 100
  - No value (`nullptr`)
- For each of the following, which value does it point to?
  - What value does the `Node` pointer `first` point to?
  - What value does the `Node` pointer `temp` point to?
  - What value does the `next` member of the `Node` with data 10 point to?
  - What value does the `next` member of the `Node` with data 100 point to?

Code Snippet 3

Assuming that everything so far has been executed, trace the following lines of code carefully to determine what memory looks like after each line.

```
5  for (int i = 1; i < 5; i++) {
6    Node* temp2 = first->next;
7    first->next = new Node((i + 1) * 100);
8    first->next->next = temp2;
9  }
```

Question 3:

- How many (non-null) `Node` values can we access after the snippet?
- How many pointers (variables OR member fields) are `nullptr` after the snippet?
- Imagine the following lines appeared after the snippet:

```
Node* save = first;
first = new Node(100);
first->next = save;
```

  - How many (non-null) `Node` values can we access after the snippet with these additional lines?

For the rest of the lab, DO NOT assume the additional three lines were run.

Code Snippet 4

Consider the following recursive function:

```
void addToEnd(Node* curr, Node* node) {
  if (curr == nullptr) {
    curr = node;
  } else {
    addToEnd(curr->next, node);
  }
}
```

Assuming everything so far has already been executed, trace through the execution of the following function call:

```
10  addToEnd(first, new Node(0));
```

Question 4:

- At the **deepest** point in execution of this snippet, how many `addToEnd` calls were made?
- How many pointers were changed by the execution of Code Snippet 4?

Code Snippet 5

Assuming everything so far has already been executed, trace through the execution of the following loop:

```
11  Node* curr = first->next;
12  while (curr != first) {
13    if (curr->next == nullptr) {
14      curr->next = first;
15    }
16    curr = curr->next;
17  }
```

Throughout these snippets, we've created the following `Node` values:

- A `Node` with data 10

- A `Node` with data 100
- A `Node` with data 200
- A `Node` with data 300
- A `Node` with data 400
- A `Node` with data 500
- A `Node` with data 0

At this point, if we were to end our program, we'd want to clean up memory. However, to be able to `delete` dynamically allocated memory, we need to be able to access it. If we can't access it to delete it, it will "leak".

For each of the node values we created, either:

- Write a statement that will access that value, starting with a pointer variable OR
- Note that it's already "leaked"

Deliverables

Answers, diagrams, and explanations for each of the 5 code snippets.

Oral Activity 2 Timeslot

If you haven't already, make sure to sign up for an oral activity timeslot! (https://docs.google.com/forms/d/e/1FAIpQLSc6P5QHzLjhQznvjoNzerq2qdlARtsZlKKcUeVez51lbUKrCQ/viewform)

Acknowledgements

Content by Adam Blank, adapted for C++ and CS 251 by Ethan Ordentlich