CS 211 Lab 5 – Summer 2023

Logistics

For this lab you will edit the file lab5.c to work with linked lists, including both statically and dynamically allocated nodes. Submit your edited program to Gradescope.

Important Notice: All programs will be evaluated on *bertvm* servers. Make sure all lab assignments are completed (or at least tested) on *bertvm* before submission.

Note also: It is not expected that all students will finish this lab during the lab period. That is fine. Turn in whatever you have accomplished within 90 to 120 minutes and move on.

Overview

This lab addresses linked lists in C. Follow the instructions and complete the given tasks in the lab5.c file. All related materials are posted on the course website under Lab Assignments -> lab5.

By the time you have completed this lab, you should be able to:

- Declare and use linked list nodes.
- Create linked lists using either statically or dynamically allocated nodes.
- Traverse a linked list for printing (or other) purposes.
- Add or remove links at the head of an existing linked list.

Instruction

Access the program lab5.c from the Lab Exercises page. Complete the tasks as outlined in this document and in the comments in the file.

Task 1: Complete the type definition for a new type, Node. The rest of this program will use instances of type Node.

Task 2: Create three Nodes as ordinary variables (not dynamically), populate them with data, and link them together into a linked list. The variable "list1" should point to the head of the list, and don't forget to set a NULL pointer at the end of the list.

Task 3: Complete the printList( ) function by uncommenting and completing the lines that are given. Print list1 and confirm that it is as expected.

Compile your program and run it at this point. Fix any errors before proceeding.
Submit to Gradescope.

Task 4: Create another Node, and pass it to addNodeToHead( ) to add it to list1. Print the new list to show that the new Node has been added.

Task 5: Complete the addNodeToHead( ) function, so that task 4 actually works.

Compile your program and run it at this point. Fix any errors before proceeding. Resubmit to Gradescope, replacing your earlier submission.

Dynamic allocation begins beyond this point.

Task 6: Create three Node * variables, and use malloc( ) to dynamically allocate 3 Nodes. Populate them with data, and link them together into a linked list. The variable "list2" should point to the head of the list, and don't forget to set a NULL pointer at the end of the list. Print the new list.

Compile your program and run it at this point. Fix any errors before proceeding. Resubmit to Gradescope, replacing your earlier submission.

Task 7: Call addValueToHead( ), passing list2 and an integer to be added to the list. Print the new list to show that a new Node has been created and added.

Task 8: Complete the addValueToHead( ) function, so that task 7 actually works.

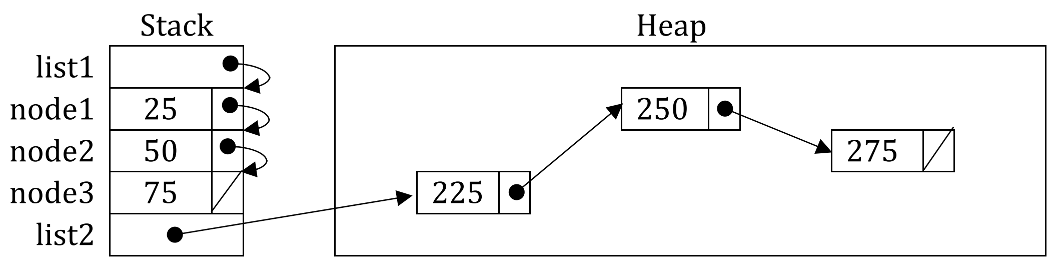
Compile your program and run it at this point. Fix any errors before proceeding. Resubmit to Gradescope, replacing your earlier submission.

Task 9: Run valgrind on the program and make note of the resulting messages. Then complete removeNodeFromHead( ) and verify that the lists shrink node by node as removeNodeFromHead( ) is called from within the loop.

Task 10: Run valgrind again, and fix any remaining problems.

Compile your program and run it at this point. Fix any errors before proceeding. Resubmit to Gradescope, replacing your earlier submission.

If properly created/allocated, these lists in memory will look like the following, (before the extra nodes are added):


Stack
Heap
list1
node1
node2
node3
list2
25
50
75
225
250
275