CS 2110 Timed Lab 3
Dynamic Memory Allocation

Preston Olds

Summer 2018

Contents

1 Before You Begin

Please take the time to read the entire document before starting the assignment. We have made some important updates, and it is your responsibility to follow the instructions and rules.

2 Timed Lab Rules - Please Read

2.1 General Rules

1. You are allowed to submit this timed lab starting at the moment the assignment is released, until you are checked off by your TA as you leave the recitation classroom. Gradescope submissions will remain open until 6 pm - but you are not allowed to submit after you leave the recitation classroom under any circumstances. **Submitting or resubmitting the assignment after you leave the classroom is a violation of the honor code - doing so will automatically incur a zero on the assignment and might be referred to the Office of Student Integrity.**
2. Make sure to give your TA your Buzzcard before beginning the Timed Lab, and to pick it up and get checked off before you leave. **Students who leave the recitation classroom without getting checked off will receive a zero.**
3. Although you may ask TAs for clarification, you are ultimately responsible for what you submit. **The information provided in this Timed Lab document takes precedence.** If in doubt, please make sure to indicate any conflicting information to your TAs.
4. Resources you are allowed to use during the timed lab:
    - Assignment files
    - Previous homework and lab submissions
    - Your mind
    - Blank paper for scratch work (please ask for permission from your TAs if you want to take paper from your bag during the Timed Lab)
5. Resources you are **NOT** allowed to use:
    - The Internet (except for submissions)
    - Any resources that are not given in the assignment
    - Textbook or notes on paper or saved on your computer
    - Email/messaging
    - Contact in any form with any other person besides TAs
6. **Before you start, make sure to close every application on your computer.** Banned resources, if found to be open during the Timed Lab period, will be considered a violation of the Timed Lab rules.
7. We reserve the right to monitor the classroom during the Timed Lab period using cameras, packet capture software, and other means.

2.2 Submission Rules

1. Follow the guidelines under the Deliverables section.

2. You are also responsible for ensuring that what you turned in is what you meant to turn in. After submitting you should be sure to download your submission into a brand new folder and test if it works. No excuses if you submit the wrong files, what you turn in is what we grade. In addition, your assignment must be turned in via Canvas/Gradescope. Under no circumstances whatsoever we will accept any email submission of an assignment. Note: if you were granted an extension you will still turn in the assignment over Canvas/Gradescope.
3. Do not submit links to files. We will not grade assignments submitted this way as it is easy to change the files after the submission period ends.

2.3 Is collaboration allowed?

Absolutely NOT. No collaboration is allowed for timed labs.

3 Overview

You will be implemented a `char` singly-linked list in C with a null terminator. This list will conform to the following requirements:

- The `LLIST` is singly-linked – each `LNODE` points only to the next node in the list
- The `LLIST` has a `head` pointer stored in the struct
- The `LLIST` has a `tail` pointer stored in the struct
- The `LLIST` has a `size` corresponding to the number of nodes in the list, including the null terminator
- When the `LLIST` is non-empty, the last node will have `data` equal to `\0`
- The last node in the `LLIST` will have its `next` pointer set to `NULL`
- The `LLIST` only contains `char` values as data, so you will store the `char` in the node without calling `malloc()` / `free()` to allocate space

3.1 Assignment Files

We have provided you with the following files:

`llist.h`	Header file for the singly-linked list (**Do not modify!**)
`llist.c`	Implement the singly-linked list in this file
`test.c`	Test your work with this file
`Makefile`	Builds the program

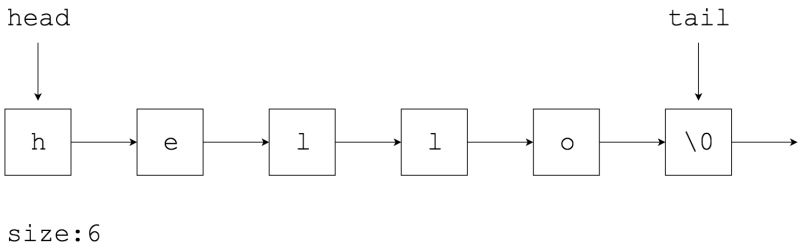
The **only** file you must edit and submit is `llist.c`, but editing `test.c` will also be helpful.

PLEASE DOUBLE CHECK YOUR SUBMISSION BEFORE YOU LEAVE!

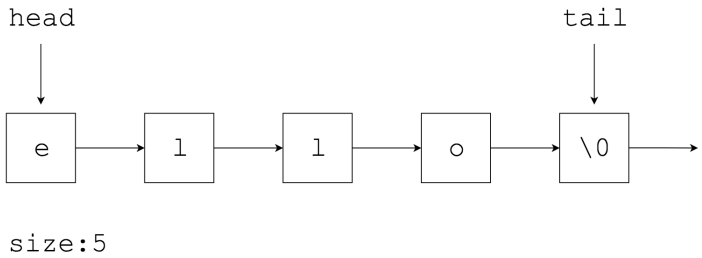
4 Instructions

4.1 `char` Singly-Linked List

Consider the string `"hello"` stored in this data structure. The following diagram depicts this example:



After calling the function `pop_front` on the list, we would have the following:



You will implement the following functions in `llist.c`:

`create_llist`	Creates a new list, initializing the fields and returning a pointer to the list struct
`create_lnode`	Create a new node, initializing the fields and returning a pointer to the node struct
`push_front`	Adds a node with the given data to the head of the list
`pop_front`	Removes the node at the head of the list
`destroy_llist`	Frees all memory in the node allocated from the heap
`get_string`	Returns a *persistent* `char*` – a string from your list's data

4.2 Restrictions

- Your code must not crash, run infinitely, or produce any memory leaks.
- Your code must compile with the `Makefile` we have provided.
- Your code must be optimal in terms of memory allocation operations (i.e. allocating more memory than needed is not optimal)

5 Testing Your Work

We have provided you with a file called `test.c` with which to test your code. Note that it contains no test cases. You must write your own.

We have provided a `Makefile` for this assignment that will build your project. Here are the commands you should be using:

1. To run the tests in `test.c`: `make run-test`
2. To debug your code using gdb: `make run-gdb`
3. To run your code with valgrind: `make run-valgrind`

6 Deliverables

When you are finished, run `make submit` and submit the following files to Canvas:

- `tl03_submission.tar.gz`