Lab 1: Linked list in C Programming

Due: Tuesday, Sept 14, 23:59

This first lab assignment is for aimed to give you a heads up on the low-level semantics of the C programming language that you will be using throughout this course.

Fork & Clone instructions

First make sure you fork the root/lab1 repository to your private namespace! If you don't, you won't be able to submit your work. Go to https://cs230.kaist.ac.kr/root/lab1 and fork to your private repo.

Once you have your private repo, you may clone your private repo. Also, **make sure you have added your SSH key to the GitLab Web UI.

```
$ git clone ssh://git@cs230.kaist.ac.kr:10022/cs[your_student_id]/lab1.git
```

1. Introduction

For this lab, your task is to fill in the skeleton code and implement the doubly linked list. You must only fill in the skeleton code in the `list.c` file. You may modify the `test.c` file and add the `iterate_print_keys()` function to help with your debugging, however you must make sure your tests work with the original `test.c` file.

2. Doubly Linked List

For this lab, you will need to implement a doubly linked list, where there are two nodes, `head` and the `tail` node that are reference nodes. They do not hold any meaningful `key` values, just `-1` as a key. The `head` and `tail` , nodes represent the beginning and the end of the list. Also the invariant of the `head` and `tail` node is that, the `tail` 's next node is the `head`, and the `head` 's previous node is the `tail`. This provides the advantage of being able to find the `tail` of the list, when given the `head`, with ease. Once you have the `head` and the `tail`, you can set your iteration to begin at the head, and end at the tail.

The functions that need to be implemented are provided in the `list.c` file. The requirements of each skeleton function is described in the block comment above the skeleton function.

3. Memory Management

You will need to dynamically allocate memory for new nodes using the `malloc()` function, and free any deleted nodes using the `free()` function. You can access the reference for the two functions using the following commands:

```
 $ man malloc
$ man free
```

4. Test cases must be met

All the test cases implemented in the `test.c` file must be met. As of this writing, there are eight test cases. Make sure to satisfy all of the test cases. You can test your `list.c` implementation using the following command

```
$ make test
```

There are three test failures that have been identified (there may be more). 1. Segmentation Fault: there is a pointer dereference in your code that is to an incorrect memory address! 2. Test Failure: some of the test cases are not being satisfied. 3. Timeout: your code is not finishing up in time. Perhaps you have a infinite loop in your code?

Once you fix all the problems, and your `make test` results in a `Passed the test, Everything worked!` output, then you should be ready to hand in your lab. Make sure your tests pass with the original `test.c` file. If you made changes to the test.c file, and committed the changes, you can revert your changes to the `test.c` file using the following command.

```
$ git checkout handout test.c
```

5. Hand in Instructions

Once you have finished your implementation and your test cases have passed, you can submit your code. Make sure that you add the `list.c` file, and commit your changes. You can do so with the following commands

```
 $ git add list.c
$ git commit -m "Your commit message"
```

Execute below command to make sure that you do not have any uncommitted changes to the `list.c` file.

```
$ git status
```

To hand in your lab, execute the following command

```
$ make handin
```

This step will push your local commits onto your GitLab remote repository. Make sure that your remote repository is your forked version of the lab1. Your remote URL should be something like ssh://git@cs230.kaist.ac.kr:10022/cs[your_student_id]/lab1.git. You can check your remote url using the following command:

```
$ git remote -v
```

Check the tags section in the GitLab Web interface to see if your latest code has been pushed onto the server. The URL to check your tags is https://cs230.kaist.ac.kr/cs[your_student_id]/lab1/tags.

6. Grading

We are going to grade only the tagged one. So, you should type `make handin` to tag the submission to your files. **The last tagged commit will be your submission date, so please be careful.** Also, if you execute it multiple times, the tag will be renewed with your new commit.

Grading Policy

- Lateness penalties: penalized 30% per day
- Only 1 day after due date is allowed