Tutorial 7
More Linked Lists

Learning Objectives

After this tutorial, you will be able to:

- Insert and delete nodes from any position in a singly linked list

Tutorial

In order to receive full marks, you must complete everything in the “Tutorial” section. Part marks may be provided if good progress was made toward completing the tutorial, at TA discretion.

In this tutorial, your functions should now return an integer representing the status of a function. There are defined status constants `C_OK` and `C_NOK` that you should use.

1. Download the file `T07.tar` from the tutorial module on Brightspace. Untar and read through the tutorial files.

2. Write a function `insertStudentAlpha(StudentList* stuList, StudentType* stu)` which takes as input a singly linked list (assumed to be sorted alphabetically by last name) and adds a student to `stuList` so that it remains in sorted order.

3. Write a function `deleteStudent(StudentList* stuList, char* fname, char* lname)` which deletes the student with matching first and last names from the list. Assume that `stuList` is sorted alphabetically by last name. Use this assumption to minimize the number of comparisons needed in the case where the student is not in the list. This function should free memory for both the node and the data. If a student cannot be found, your function should return `C_NOK`.

4. Write a function `cleanupList(StudentList* stuList)` which frees all memory associated with `stuList`.

Exercises

These exercises are optional, but are provided to give you some additional practice working with common operations that you will encounter while using pointers that might be tricky when you first work with them.

1. Write a function `sortList(StudentList* stuList)` which takes an unsorted singly linked list and returns the list in alphabetically sorted order.

2. Write a function `gpaRange(StudentList* stuList, StudentList** result, int minGPA, int maxGPA)` which returns (via `result`) a new linked list which contains all the students in `stuList` having GPAs greater than or equal to `minGPA` and less than or equal to `maxGPA`.

Tutorial 7 - Linked Lists

Saving and Submission

In-Person Attendance: Submission is optional, as you can be checked off in class, but you must still complete the required portion of the tutorial for marks.
Asynchronous: If you are completing the work on your own at home, make sure to follow the submission requirements.

1. Create an archive to submit your files, and include all of the files that you worked with in the tutorial.
    a. They may be submitted as a .tar, .tar.gz, bz2, or .tgz file
    b. Make sure to include all of the code needed to run your program

2. For **full marks** you should have completed all problems in the Tutorial section (and they should be seen upon running the program) without any memory leaks on exit. The exercises are optional, but highly recommended for the extra challenge and exploration of these concepts.

3. For **part-marks** you should have attempted to complete most of the tutorial. Grades for part-marks will be at TA discretion.