Homework 3:
Grade Database Management Report

Submitted: November 10, 2017

Sabbir Ahmed

1 Description

This project simulates a simple student and grade database management system using linked lists.

The program will read in a text file of students and their grades. Each student will have their own node in a linked list containing the student's name, a list of their grades and their grade average. A user will be able to search for a student's name via the command line. The user will also be able search a grade and to list out all students with that grade. Along with being able to search for a student or grade, the user will also be able to add and remove students from the database.

This document serves as the final report for the project. The report details the implementation of the project using C in a full Linux environment. No AVR libraries or environments were used in the development.

2 Implementation

The project was developed with a bottom-up design. Functions with more frequent usage and higher priority were developed first, and later pieced together to contribute to higher level functionalities.

2.1 database

The nodes and linked list containers making up the database were developed first. Basic functionality of the linked lists, such as adding and removing nodes were developed in the initial stages, followed by an iterator wrapper to help print out the contents of the linked list.

The concepts of using iterators and doubly-linked lists were influenced from `Clibs` (1). No code was borrowed from their implementation. Using a doubly-linked list allowed efficiency in the `db_remove()` method. Linking the previous node of the current node queued for removal could be referenced in constant time; whereas a singly-linked would have to iterate through the entire linked list to reach the previous node of the current node.

The sorting functionality was implemented using a simple bubble sort algorithm modified from the snippet provided by `GeeksforGeeks` (2).

2.2 list_of_grades

The nodes and linked list containers making up the `grade_t` structures were developed later. `grade_t` was implemented as a struct unionizing the two `quiz_t` and `test_t` structs.

2.3 Memory Management

`valgrind` was used to detect and track memeory leaks of the program. An essential rule of thumb was developed, where each instances of `mallocs` were followed by `free` statements after usage. All memory leaks have been accounted for in the current version of the program.

3 Usage

The project depends on user inputs from the terminal. Once executed, the Main Menu is displayed, prompting the user for a choice of 6 options: `Print Database`, `Search by Name`, `Search by Grade`, `Add Students`, `Remove Students` and `Exit`. All these options, excluding `Print Database` and `Exit`, runs on independent loops until the user decides to exit.

3.1 Print Database

The `Print Database` option displays all the students in the database, along with their assignment and final course grades. The database is initialized if an input file with valid student information is provided. If no input file is provided, the database is initialized as an empty linked list.

3.2 Search by Name

The `Search by Name` option allows the user to search for a student and their grade information with their last name. The user is prompted for the student's last name, and their input is compared case-insensitively with the last names of all the students in the database.

3.3 Search by Grade

The `Search by Grade` option allows the user to search for a student and their grade information with their letter grade. The user is prompted for the their desired letter grade (A, B, C, D, or F), and their input is compared with the final grades of all the students in the database. The program will print out every instances of student final grades matching the user input.

3.4 Add Students

The `Add Students` option allows the user to add new students to the database. The user is prompted for the new student's name, and then given the option to select the type of the new grade. The types include quizzes and tests. Once a grade type has been established, the user gets prompted for the quiz / test name, grade and weight. This process runs in a loop until the user is satisfied with the student information.

3.5 Remove Students

The `Remove Students` option allows users to remove students from the database using their last names. The user is prompted for the student's last name, and their input is compared case-insensitively with the last names of all the students in the database. Once a match has been established, the node is freed from memory and the student is removed from the database.

3.6 Exit

The Exit option breaks out of the loop and immediately terminates the program.

3.7 Invalid Choices

The program will continue displaying the Main Menu until a valid choice is inputted by the user.

3.8 File I/O

Providing an input file as a command line argument is optional. If a path to an input file with valid data is provided, the program will parse the input to initialize the database with those students. The number of students in the input file must be predetermined for the program to properly parse the data. The number of students may be modified with the `STUDENTSINFILE` constant. The constant is currently set to `2`. If no input file is provided, the database will be initialized empty.

4 Testing and Troubleshooting

Managing memory leaks proved to be much more challenging than the actual implementation of the project. `<strings.h>` functions consisted of several

constraints regarding the size of input string buffers, locations of the NULL terminator, etc. Once all the memory leak was taken care of, the code was frozen and documented. In the final stages, multiple test cases were considered in validating the functionality of the program such as duplicate last names being properly sorted, invalid letter grades as inputs, etc.

5 Code

The C scripts used for the implementation has been attached alongside the report.

5.1 main.c

Driver file for Homework 3: Grade Database Management.

5.2 menu.c

Contains the implemention of the functions used to provide high level interfaces to the data structures in the project.

5.3 database.c

Contains the implementation of student_t nodes and database_t linked lists constructors, methods and its corresponding iterators.

5.4 list_of_grades.c

Contains the implementation of grade_t nodes and list_of_grades_t linked lists constructors, methods and its corresponding iterators.

5.5 Makefile

A Makefile has been provided with the project with the following recipes.

- `build` : compile the program with GCC
- `run INPUT=path/to/file.txt` : run the GCC-compiled executable. If the optional `INPUT` arugment is not used, the database is initialized empty.
- `val INPUT=path/to/file.txt` : run `valgrind` on the executable. If the optional `INPUT` arugment is not used, the database is initialized empty.
- `clean` : remove temporary and executable files

References

(1) Clibs Authors. "Package manager for the C programming langage." *clib(1)*. Accessed November 10, 2017, *http://clibs.org/*.

(2) GeeksforGeeks. "C Program for Bubble Sort on Linked List." *GeeksforGeeks*. Accessed November 10, 2017, *http://www.geeksforgeeks.org/c-program-bubble-sort-linked- list/*.