COMP 2160 Programming Practices
Assignment 3

Due Date: Sunday, November 19th at 11:59 pm

Objectives

- Separate compilation, makefiles
- Abstract Data Types

Question 1

Take the sample solution from Assignment 2 and factor the linked list code into a **Table** (implemented internally using a Linked List). To do this, create `Table.c` and `Table.h` files, with the solution from Assignment 2 providing the implementation of `insert(), firstItem(), nextItem(), search(), clearTable()` and `validateTable()` (called `clearList()` and `validateList()` in the sample solution). You are also required to add a `delete()` function (which deletes the first occurrence of an item).

To support the use of multiple Tables within an application, your Table module must also include the ability to manage multiple tables. Create a struct to contain the 'instance' variables that describe a Table. Add routines `createTable()` and `destroyTable()` (which uses `clearTable()`...) and modify all other routines so that they take a table pointer as their first parameter (they need to know which table to work with -- note that this is how the 'this' pointer is implemented in OO languages such as Java and C++). Note that anything you add (including, but not limited to, the new routines) **must** adhere to the principles of Design by Contract, as per the requirements in Assignment 2.

Finally, implement a `main.c` program that runs a suite of tests (unit tests) to ensure that the Table correctly handles insertions, removals, searches and traversals. Your tests should focus on the operations on a table, not the manipulation of multiple tables. Be sure to factor your tests into logical units that are implemented as routines. You must include textual output indicating each test performed (e.g. "Searching for elements not in the table:") and whether or not they passed (e.g. "george found" or "fred not found"). Don't forget to run tests on an empty table. Upon completion, include a summary of the number of tests run, the total number that passed and the total number that failed (note that with assertions off you should always run to completion and get this summary).

Requirements

- You must include a Makefile for compiling your complete program.
- Make sure you test with assertions turned off. Boundary conditions should not cause the program to crash.
- You cannot change the internal behaviour of the Table (as defined by the sample solution from Assignment 2).
- You cannot use your own solution to Assignment 2. Learning to work with other people's code is very important.

Hand-in Instructions

Go to COMP2160 in UMLearn, then click “Assignments” (under “Assessments” menu). You will find a assignment submission folder called “HW3”. Click the link and follow the instructions. Please note the following:

- Submit ONE zip file containing your source codes(both .c and .h files), Makefile, and a README (README.txt) containing the summary of the number of tests run, the total number that passed and the total number that failed Although not required. You may also explain anything unusual about compiling/running your program in the README. Please name the zip file as `A3<your last name><your student id>.zip` (e.g. `A3Seo1234567.zip` or `A3Wang7654321.zip`).
- You may resubmit your assignment, but ONLY THE LASTEST submission will be kept.
- We only accept homework submissions via UMLearn. Please DO NOT try to email your homework to the instructor or TAs.
- We reserve the right to refuse to grade the homework or to deduct marks if these instructions are not followed.