CIS*2750 Assignment 2 evaluation

1. Grading scheme

Module 1

writeGEDCOM	25%
validateGEDCOM	5%

Module 2

getDescendantListN	22%
getAncestorListN	22%

Module 3

indToJSON	4%
JSONtoInd	5%
JSONtoGEDCOM	7%
addIndividual	2%
iListToJSON	4%
gListToJSON	4%
Total:	**100%**

There will be no new Makefile requirements for Assignment 2, and your Makefile will not be graded. However, make sure you use the correct directory structure for your submission. See Section 4 for details.

2. Upgrading your Assignment 2 grade:

You will have an opportunity to make a minor upgrade to your A2 grade by resubmitting it The resubmission procedure will work the same as A1: there will be a dedicated dropbox, and the deadline for A2 resubmission will be the same as A3 deadline.

The final Assignment 2 grade will be calculated using a weighted average as follows:

A2 final grade = 0.7*initial A2 grade + 0.3*resubmitted A2 grade

NOTE: You **must** submit A2 to be able to upgrade it. So submit A2 even if you think you'll get a zero for it - you'll be able to bump up your grade a bit later. However, if you do not submit it, your grade will be 0 and the upgrade option will not be available to you.

3. Evaluation notes:

- If you are aiming for part marks, you do not have to implement every single required function. However, you **must** include a stub for every single function listed in Modules 0 - 3. Without the stubs, your submission will not compile with the test harness, and will automatically get a grade of 0.

- writeGEDCOM() will be evaluated by saving a reference GEDCOMobject to a file, reading the file into another GEDCOMobject, and comparing two GEDCOM objects. As a result, details like your XREF naming conventions and the order in which you save Individuals/Families to file do not matter. However, your createGEDCOM() must function correctly. The "Harsh" A1 test harness will help you

verify that createGEDCOM() works as expected.

- The functions that accept GEDCOM objects as arguments will be tested on reference GEDCOMobjects, as well as GEDCOMobjects created from files using your createGEDCOM(). They must not modify the argument GEDCOM objects in any way.

- Make sure that you get the format of the output JSON strings right - they will be directly compared to reference strings.

- A very simple public A2 harness will be posted to make sure you didn't make any really obvious mistakes, e.g. forget to add stubs for unimplemented A2 functions. It will include 1 or 2 very simple writeGEDCOM() tests, as well as a couple JSONtoSomething()/SomethingToJSON() tests. It will be posted by the end of the weekend, a couple of days before the A2 deadline.

4. Deductions

Your code must compile, run, and implement all functions listed in Modules 0 - 3. As with A1, your code will be compiled and tested by the test harness using `-Wall -g -std=c11` flags. Any compiler errors will result in an automatic grade of **zero** (0) for the assignment.

Your submission must not contain any files with a main() function, or anything else that is not required by the assignment. All of the files you submit will be compiled with the test harness, so do not include anything that you do not want to be graded.

Make sure you compile and run the assignment on the SoCS Linux server before submitting it - this is where it will be graded.

You will lose mark for run-time errors and incorrect functionality. Additional deductions include:

- Any compiler warnings: -15%
- Any memory leaks: -15%
- Incorrect directory structure: -5%
- Any additional failures to follow submission instructions: -5%

5. Submission

The submission must have the following directory structure:

- `assign2/` contains the README file and the `Makefile`
- `assign2/bin` should be empty, but this is where the `Makefile` will place the static lib files
- `assign2/src` contains `GEDCOMparser.c`, `LinkedListAPI.c`, and `GEDCOMutilities.c` (if you need it).
- `assign2/include` - contains `GEDCOMparser.h`, `LinkedListAPI.h`, and any additional header files that you might have.

Submit your files as a Zip archive using CourseLink. File name must be be A2FirstnameLastname.zip.

Late submissions: see course outline for late submission policies.