CIS*2750
Assignment 3
Deadline: Friday, March 16, 11:59pm
Weight: 12.5%

1. Description

In this assignment, you will create a web-based graphical user interface (GUI) for manipulating the genealogy information stored in GEDCOM files. The front end (GUI) will be a web client written in HTML/JavaScript. The backend will use Node.js platform as the server, and will call your parser library created in Assignments 1 and 2 using the FFI Node.js module.

This assignment is individual work and is subject to the University Academic Misconduct Policy. See course outline for details.

This assignment will be released as a series of modules, instead of all at once, to facilitate incremental development. I strongly encourage you to start working on Module 1 as soon as it is released.

Module releases will be announced on CourseLink. Each module will be described in a separate document. The last module will be posted by March 9.

You will also be provided with an A3 Stub, which will contain stubs for both server and client code. You will need it to develop and run your assignment, but you can get started on the HTML portion of Module 1 without it.

If you want to work from home, you must install Node.js 8.9.4 on your home machine. It is already installed for you on the SoCS machines.

The A3 Stub will include a script to install all the Node.js libraries you will need for the server code. A3 Stub documentation will describe how to use the stub. In addition, the A3 Stub documentation will describe your Assignment 3 submission structure and Makefile details.

2. Modules

Module 1 will describe the user interface functionality for the client code running in the browser.

Module 2 will describe the server-side functionality that will interact with your parser library written in C, and communicate with the web client.

3. Evaluation

Your code must compile, run, and have all of the specified functionality implemented. Any compiler errors will result in the automatic grade of **zero** (0) for the assignment. Make sure you compile and run the assignment on the SoCS Linux server before submitting it - this is where it will be graded. See Modules 1 and 2 for details.

An evaluation scheme will be posted in Assignment 3 dropbox.

If Module 1 is implemented, but not connected to Module 2, you will get some part marks (approximately 50% of the overall assignment grade). Make sure you implement Module 1 even if you are having trouble with Module 2. However, Module 2 functionality must be connected to a functional Web app GUI described in Module 1. Module 2 functionality that is not connected to a Module 1 GUI will be worth zero (0) marks.

There will be **no** resubmissions for Assignments 3 and 4. As announced in class, you will have an option of selecting a 2-day extension either for Assignment 3 or Assignment 4. Details on this will be announced in class and posted on CourseLink.

Starting from this assignment, all evaluation will be done through your GUI. Unlike the previous assignments, the lower-level modules will not be individually tested. In general, you can obtain marks for correct front-end GUI

(HTML/JavaScript) functionality and correct back-end functionality (JavaScript/C code), or for only the GUI in case the server back-end doesn't work.

Front-end functionality means that the GUI elements are present and can be manipulated. For example, if the back-end of getting a list of descendants does not work yet, you should at least display a message that proves that the GUI element for the descendant list is connected to a callback function for the Get Descendants button. See Module 1 for details. Passive/static GUI elements with no callbacks will not get full front-end credit.

When A3 Stub is released, each of you will be assigned a unique port number (plus a backup number). This port number will be used only by your assignment. Failure to use port numbers as specified in A3 Stub documentation will result in penalties, up to and including receiving a **zero (0)** for the assignment. Connecting to someone else's code (client or server), either on purpose or accidentally, will constitute Academic Misconduct and will be treated as such.

4. Bonus marks (up to 10%)

You can get a bonus mark (5% of assignment grade) for a particularly attractive GUI. This will be determined solely by the TA and will be unavoidably subjective. If you are not comfortable with this, please do not spend any time prettifying your GUI.

Another bonus is available for providing additional information about an individual in the GEDCOM View Panel as described in Modules 1 and 2 (5% of the assignment grade).

5. Incremental development

A suggested road map would go like this:

1. Familiarize yourself with the Web UI development: HTML, JavaScript, and jQuery. Read Week 7 lecture notes, review the posted Week 7 code examples, and use *"Learning PHP, MySQL, JavaScript, CSS & HTML5*" as a reference. Chapters 13 - 15 are good starting point, since they cover JavaScript and HTML.
2. Design your GUI on paper.
3. Convert your GUI design to static HTML code that you can run in a browser.

   You can do steps 1-3 without the use of the A3 Stub.

4. Get your program working so it successfully displays your GUI. Populate it with enough fake component data to prove that the scroll bars are working.
5. Create and connect all your JavaScript callbacks, so if a user clicks on any of the buttons, appropriate output is displayed in the Status Panel.
6. **Test** this on the SoCS linux server using NoMachine using Firefox installed there (see Module 1 for details).

Do all this before you start working on Module 2. Remember, Module 2 is not worth any marks without a working GUI.

The overall functionality of Module 2 should be fairly obvious from the description in Module 1. However, Module 2 description will include the necessary implementation details and guidelines.

Once Module 1 is complete:

7. Shift your attention to the server side: design and implement any additional C functions that you might need, implement them as stubs that print out their arguments when called, and pass back some dummy data to server-side JavaScript code. Learn how to link them into a shared object library (makefiles help - use my posted example).
8. Change your JavaScript server code so it actually calls these functions, and test this until it works reliably. You have now established two-way communication between JavaScript and C
9. Modify your wrapper functions so they do their intended job. Now you can read real GEDCOM files into your server code
10. Complete the callbacks for client Ajax requests, so you can pass the JSON information between the server and the GUI.

11. **Test** this on the SoCS linux server using NoMachine using Node.js and Firefox installed there (see Module 1 and 2 descriptions for details).

6. Submission

Submit all your C and JavaScript code, along with the necessary Makefile. File name must be be `A3FirstnameLastname.zip`. The details in the assignment submission structure will be posted in the A3 Stub documentation.

Late submissions: see course outline for late submission policies.

This assignment is individual work and is subject to the University Academic Misconduct Policy. See course outline for details)