### SER421 Online: Web Applications and Mobile Systems Fall 2016B

### Lab 2, due Thursday, 10/27/16 at 11:59pm via online submission to Blackboard

***You MUST do this lab in pairs but you are required to participate and understand all parts of the solution.***

The goal of Lab 2 is to get you working in NodeJS, understanding the event queue, and the file APIs. Submission instructions are at the end, PLEASE FOLLOW SUBMISSION INSTRUCTIONS!

**Activity: Implement your own Eliza!**

Since the early days of computing, humans wondered if computers could be made "intelligent". In the AI subfield of natural language processing, the argument was presented that if a computer could communicate like a human, then it possessed human intelligence. To demonstrate this philosophical approach, a computer program named ELIZA was created to emulate a computer talking to a person by responding to queries (<https://en.wikipedia.org/wiki/ELIZA>, <http://psych.fullerton.edu/mbirnbaum/psych101/Eliza.htm>). In essence, Eliza was Siri before Siri was! For this lab I want you to implement your own Eliza!

The basic approach behind Eliza is to focus on keywords or substrings that a user presents to Eliza, and use it to determine a pseudo-intelligent response from a dictionary of responses. Further, the program "remembers" previous answers the user has presented and also applies some randomness to avoid deterministic and/or repeated responses.

Create your own Eliza under the following constraints:

1. Your keyword/phrase mappings to one or more appropriate responses should be stored in one or more JSON files. You must read in the JSON file(s) at the start of your program (10 points for this read routine).
   1. Note I am ambiguous on purpose about file or files as I leave it to your design to determine whether you should have one large dictionary file or many smaller ones.
   2. I also leave the JSON structure to your imagination. I do expect that it will have a readily identifiable keyword or phrase that maps to possible response phrases. But beyond that it is your design.
   3. I am fine if the class wants to share JSON dictionaries, as the size of the dictionary improves the perceived intelligence of Eliza. But no sharing of javascript code to work with shared JSON of course.
2. You have to be able to read input from the command-line. You may use the readline module for this <https://nodejs.org/api/readline.html>

Functionally your program must:

1. (5) Greet the user by asking for her/his name on startup. The name should be "remembered" and used anywhere a direct naming of the end user is appropriate, such as in #5 and #8.
2. (5) Eliza should start the conversation with a question. The question is not fixed, but should start with the common greetings such as "how is your day going?" "is something troubling you?" or "you seem happy, why is that?"
3. (15) Using readline, you should parse the string the user types in and search for matches in the dictionary and select one for a response.
4. (15) You should attempt to vary the responses to the same keywords (although the number of responses will of course be finite). Vary responses by
   1. Remembering responses to the same keyword in that session
   2. Introducing some simple randomization so no 2 sessions follow the same pattern of responses. For randomization, use the basic Math.random() built into Javascript.
5. (10) If the user does not respond to an Eliza question within 20 seconds, Eliza should display a prompting-type message such as "<name>, I'm waiting here!" or "Whatsa matter <name>, cat got your tongue?" or so forth (have some fun with it, but it must include <name>, where <name> is the end user's name). Again the prompt should not always be the same.
6. (5) To exit Eliza the user types "quit" by itself for input.
7. (15) Your program should detect the presence of a new JSON dictionary file in the working directory, and have the ability to dynamically incorporate the new dictionary by adding (not replacing) its entries to the existing dictionary Eliza is using.
   1. When this happens, Eliza should proudly announce: "I just got smarter!"
   2. This feature must be implemented through a custom event emitter.
8. (10) Every 3 minutes, Eliza should ask "You sure can talk. I need some coffee – join me at Dunkin, <name>?" where <name> is the end user's name. (Yes, I am a Dunkin person, you may sub in Starbucks, or Peet's or Seattle's best or your fav – but remember Eliza is supposed to be intelligent!). If the end user answers immediately with "maybe" then turn off subsequent 3-minute prompts.
9. (10) Eliza should support a "log" command. When the user enters "log" by itself on the command line, a file name "<name>\_<datetime>.log" should be created that records the entire conversation to that point in the session.

**SUBMISSION INSTRUCTIONS (READ CAREFULLY and ASK QUESTIONS!):**

1. Create a zipfile named <asurite1>\_<asurite2>\_421Lab2.zip where <asurite> is your ASURITE id. We only need one submission between the 2 students.
2. Please name your javascript file "eliza.js". It should be runnable using "node eliza.js". No package.json is required or desires.
3. Please include a file named README.txt with any information you want us to know. For example, there are some design decisions in this lab, and you can use this file to explain your design rationale. Please also indicate your partner in this file. Also indicate your partner in the Blackboard submission text box.
4. Make sure you include a dictionary file or files. Please describe your JSON format in your Readme.txt so we can create a sample dictionary to test feature #7. If you are nice you can even provide a few extra files yourself ☺
5. I strongly suggest, especially on programming problems, that you get a stable solution to a feature, save it, and then move on. You can also give us a partially done feature which we will evaluate for partial credit, but your README.txt has to tell us that (for example, include a 2nd file elizaPC.js and tell in the README what feature(s) it partially addresses)!
6. We allow as many submissions as you want to do, we only grade the last one! No reason to be late. Nothing after 11:59:00PM!
7. Remember, my expectation is that you are pair programming, not divide-and-conquer. If I find evidence to the contrary I reserve the right to give you half credit or an additional quiz or both!

**Hints and Guidance on Grading:**

* Functional programming lends itself very very well to "Test a little, code a little"!
* I am fully aware that there are Javascript and even Node-specific examples of Eliza out on the Internet. I remind you that using this code, even looking at it, is an Academic Integrity Violation (and besides, I have looked and none I can find are done under the constraints I have specified for you anyway!).
* Do not submit poorly constructed code. You are at a level of training and maturity where handing something in hacked at 11:58pm is a poor process. Start early, finish early, and allocate time to refactor and clean. Be proud of the code you submit! I reserve the right to deduct points at my discretion for poor code.
* I hope you have watched the video on course policies regarding academic integrity, ethics (working in pairs), and late policies. I believe I was very clear in this material – written and video. If you haven't yet, no time like the present!