Programming Assignment #1 CSC 4240/5240
Due: Friday September 22 BEFORE class

Description:

For this programming assignment, you will be creating a personalized version of an Eliza system using LISP. You will be given source code to use as the basis of your project and then you must add components and extend the given components as discussed below.

The original ELIZA program was published by Joseph Weizenbaum in 1966. It simulated a Rogerian therapist; that is, most of what it said was merely repeating (with appropriate substitutions) what it was told. Here is a bit of sample dialogue (from Rich & Knight, *Artificial Intelligence*, 1983):

Person: Men are all alike.
ELIZA: In what way?
Person: They're always bugging us about something or other.
ELIZA: Can you think of a specific example?
Person: Well, my boyfriend made me come here.
ELIZA: Your boyfriend made you come here.
Person: He says I'm depressed much of the time.
ELIZA: I am sorry to hear you are depressed.
Person: It's true. I am unhappy.
ELIZA: Do you think that coming here will help you not to be unhappy?
Person: I need some help, that much seems certain.
ELIZA: What would it mean to you if you got some help?
Person: Perhaps I could learn to get along with my mother.
ELIZA: Tell me more about your family.

ELIZA worked by applying pattern-matching rules. It looked for a rule whose left side matched the last sentence entered by the user, then used the right side of that rule to generate a response. A sample rule might be something like (I REMEMBER X) => (WHY DO YOU REMEMBER X JUST NOW).

For this project, you will be given a starter LISP file that implements Eliza on a sentence-by-sentence basis. If you load this file into the CLISP environment (e.g., >clisp –i eliza_starter.lisp), you can run it like this:

```
> (eliza '(hello))
(HELLO - HAVE A SEAT AND TELL ME HOW YOU FEEL TODAY.)
```

Eliza responds with a response from the meager database provided. Your task is to write rules that make Eliza seem human, at least for a minute or two. In particular, you must make Eliza sound like YOU in conversation. To do this you must first expand the database to include more responses than those provided for you. Interactions that work best are the heavily scripted ones. Write the rules carefully. In general, it is quality and not quantity that counts.

Secondly, expand the function ‘change-pros’ in order to fit with the kinds of responses you are trying to evoke.

Guidelines:

- Download the file `eliza_starter.lisp` from the class (iLearn) website. MAKE SURE IT WORKS FIRST BEFORE YOU MAKE ANY CHANGES!
- Eliza switches pronouns in the sentence so that when she matches a phrase to repeat it back to you, "I think I am an idiot" can become "You think you are an idiot." Extend the '`change-pros`' function to cover additional situations other than "I" to "you" and vice versa. You may want to think about possessive pronouns, such as "my" and "your."
- Extend the `database` to include more responses. Below is a discussion on how the '`instantiate`' function works with the pattern-matching.
- Extend the rules to recognize certain keywords and respond to those. For example: "I think I'm in **love**" => "Love isn't always a simple thing."
- Provide general-purpose responses which will be used if no other patterns match, for example: "Tell me more about that." There should be more than one such response, selected randomly.
- Log a good conversation with at least 10 unique responses to sentences beyond what is in the starter file.
- Log a bad conversation with at least 10 unique poor responses. Discuss briefly (1-2 paragraphs) why your implementation fails on these responses.
- Do *not* worry about (1) capitalization, (2) punctuation, or (3) parentheses. It's traditional in LISP programs to see input and output such as (TELL ME MORE ABOUT THAT).

Partitioning a Sentence:

To understand the database, look at one of the entries provided:

```
( (0 you think 0)
(And just why do you think 4 ?) )
```

The number '0' matches against any number of words and partitions a sentence in the following way:

...	you	think	...
1	2	3	4

The phrases matched to the number '0' are in positions 1 and 4. This is why the response uses the number 4; it is a signal to take whatever the user said that matched that partition and repeat it back at the end of the response. When you partition your sentence similarly, use '0' for phrases of zero or more words, and consider each of those phrases matched against one slot in the partition for use in the response.

Note that the 'change-pros' function is run on the sentence BEFORE the pattern-matching. This is so that the sentence 'I think I am an idiot' does not result in the response, 'Why do you think I am an idiot?' You need the pronouns changed first. This way, your database entries could always assume the pronouns have already been switched.

Submission:

Put *documentation*, *logs*, LISP source *code* and *analysis* for your solution into the class website (iLearn) dropbox, in the folder labeled "Program One" no later than on the specified due date and time. THE DROPBOX WILL BE LOCKED AFTER THAT, AND NO LATE SUBMISSIONS WILL BE ACCEPTED.

Note: that there are many Eliza versions on the web and your code will be compared against them to insure against plagiarism.

Code requirements: (40 points)

- The code should be completely working, and thus must include both what you are given AND the code that you wrote.
- Your rules.

Documentation: (10 points)

- A short (1 page or less) documentation file named README describing how to use your program.

Logs: (20 points)

TWO different transcripts of your Eliza program running, with annotations if necessary. In these transcripts you must log TWO conversations with your version of Eliza as described below:

- one in which she comes up with at least 10 unique responses other than the ones provided – and she makes sense
- one in which there are at least 10 responses which illustrate Eliza's shortcomings as a pattern-matching conversation agent

Analysis requirements: (30 points)

- A paragraph on what you think you did well in the design of the rules. What was interesting?
- A paragraph on what aspects of your implementation result in the shortcomings demonstrated in the "poor" conversation transcript.
- A paragraph on how Eliza should be extended.

ALL files should be plain text (unformatted) files.

Additional Notes:

- This is an upper-level/grad course. I expect good software engineering principles.
- Read instructions carefully
- README files should be user-friendly. Look at some Unix MAN pages for examples.
- Analysis should be in-depth
- NEVER use other code unless instructed OR cleared by the instructor