Homework 4
CS161, Winter 2016
Due: Friday, March 11 before 11:55 pm. Submit via CCLE.

This is your last homework!

You are going to write some functions useful for generating English when given conceptual answers to conceptual questions. Such functions would be part of a more general Q/A system for your story comprehension system. We will assume that your Q/A system has already run demons to analyze a question and turn it into a conceptual question (i.e. a concept that represents the meaning of the question). We will also assume that your Q/A system has already retrieved a conceptual answer by means of backward chaining (you will not have to implement this backward chainer). Thus, all you have to define are a couple functions involved in generating English sentences when given conceptual answers.

Here are several questions (Q-CONX) and answers both in frame (C-ANS-X) and English (ENG-X) formats:

[E] Example 0.1: "What does GEORGE do at the high school?"

```
Q-CON1 =
(ACT AGENT (HUMAN F-NAME (GEORGE)
                 GENDER (MALE))
     LOC (HS REF (DEF))
     CLASS (V WHAT?))
```

Given the concept of this question Q-CON1, we assume your Q/A system has already returned the following conceptual answer:

```
C-ANS-1 =
(TEACH AGENT (HUMAN F-NAME (GEORGE)
                   GENDER (MALE))
       RECIP (HUMAN TYPE (STUDENTS))
       OBJECT (CHEM)
       LOC (HS REF (DEF))
       SITU (S1))
```

We would like to have a means of translating the frame-conceptual answer above (C-ANS-1) into one that is readable by a human (ENG-1, below):

```
ENG-1 = (GEORGE TEACHES CHEMISTRY AT THE HIGH SCHOOL TO STUDENTS)
```

[E] Example 0.2: "Why did lesions show up on Rick's nose?"

```
Q-CON2 =
(CAUSE ANTE (V WHY?)
       CONSEQ (APPEAR OBJECT (LESIONS LOC (NOSE PARTOF
                                              (HUMAN F-NAME (RICK)
                                                     GENDER (MALE))))))
```

```
C-ANS-2 =
(INGEST AGENT (HUMAN F-NAME (RICK)
                     GENDER (MALE))
        OBJECT (COCAINE)
        TO (NOSE PARTOF (HUMAN F-NAME (RICK)
                               GENDER (MALE)))
        SITU (S4))

ENG-2 = (BECAUSE RICK PUT COCAINE INTO NOSE OF RICK)
```

[E] Example 0.3: "How did George find out he is ill?"

```
Q-CON3 =
(ENABLES ANTE (V HOW?)
         CONSEQ (KNOWS AGENT (HUMAN F-NAME (GEORGE)
                                    GENDER (MALE))
                       OBJECT (STATE AGENT (HUMAN F-NAME (GEORGE)
                                                  GENDER (MALE))
                                     OBJECT (DISEASED))))

C-ANS-3 =
(INFORM AGENT (HUMAN ROLE (ONCOLOGIST)
                     REF (DEF))
        RECIP (HUMAN F-NAME (GEORGE)
                     GENDER (MALE))
        OBJECT (STATE AGENT (HUMAN F-NAME (GEORGE)
                                   GENDER (MALE))
                      OBJECT (CANCER TYPE (TERMINAL)))
        SITU (S3))

ENG-3 = (THE ONCOLOGIST TOLD GEORGE THAT HE HAS TERMINAL CANCER)
```

You will now define two Lisp functions that will aid in the task of translating these frames into human-readable English sentences:

1. FR-TO-ENG will look for translations based on predicates and slot-names, with possible ambiguity resolution performed by a *decision tree*.
2. EVAL-D-TREE will perform the decision tree parsing based on the input answer and question concepts.

Logically, a *decision tree* consists of a set of *paths* where each path leads to a leaf in the tree. These paths are ORed together. That is, only one path need be true. Each *path* consists of a set of tests that are ANDed together; i.e., <u>all</u> of the tests must be true (non-NIL) to reach the leaf at the end of a path in the tree. Each leaf in the decision-tree (in this homework) will contain an alternative translation pattern to be used. So when a decision-tree is encountered, it results in a pattern that **FR-TO-ENG** will then use.

A path in any of our decision-trees will be of the form:

```
(AND test1 test2 … test-n new-pattern)
```

Therefore, a decision-tree consists of a list of paths:

```
D-TREE = (
  (AND test11 … test1x alt-pattern1)
  (AND test21 … test2x alt-pattern2)
              …
  (AND testk1 … testkx alt-pattern-n)
)
```

We will discuss decision trees further in Problem 2, with examples to illustrate their usage.

Problem 1: (FR-TO-ENG e-pats c-ans q-con)

FR-TO-ENG is our English-translation engine, which will translate the given answer frame (c-ans) into its English equivalent. To do so, it takes three parameters:

1. **E-PATS:** an association list that contains predicate-translation elements of the form:
   `(PRED pattern+)`
   where PRED is a frame-predicate to which the given pattern(s) will apply, and:
   a. A *pattern* is a list consisting of phrases, slot-names, and *decision trees*.
      i. A **phrase** consists of a list beginning with the keyword PHRASE followed by a literal sequence of English words that are replicated *exactly* in the English translation.
      ii. A **slot-name** item consists of a list beginning with the keyword SL-NAME, and will recursively call FR-TO-ENG with the filler of the provided slot-name in c-ans; whatever that recursive call returns will be placed at the position of the given slot-name element in the translation pattern. Note: if a slot-name does *not* exist in the current c-ans, then the slot-name pattern will append NIL to the final result (i.e., it will do nothing).
      iii. A **decision-tree** consists of a list beginning with the keyword D-TREE and will evaluate the given decision tree (assumed to be global and well-defined at the time of execution) on the provided c-ans, and return a replacement-pattern to be used at the position of the decision-tree item in the given pattern.
2. **C-ANS:** an instantiated answer-frame (like C-ANS-1 above).
3. **Q-CON:** an instantiated question-frame (like Q-CON1 above) used for D-TREE resolution.

[!] NOTE: When **FR-TO-ENG** is first called, it assigns its arguments to 3 *global variables*:

- *CUR-Q-CON is assigned the value of **Q-CON**
- *CUR-C-ANS is assigned the value of **C-ANS** *the first time that FR-TO-ENG is called, i.e., *CUR-C-ANS is NOT set on recursive calls internal to FR-TO-ENG*.
- *CUR-FILLER is also initially assigned the value of **C-ANS**. When **FR-TO-ENG** calls upon itself recursively to generate English for a given filler, then **FR-TO-ENG** will assign *CUR-FILLER to be that filler. You might consider an optional parameter to help decide when to set *CUR-C-ANS and when to set *CUR-FILLER!

So, FR-TO-ENG takes in a list of English patterns, an instantiated answer concept, and an instantiated question concept, and then returns the translated sentence. First, here is an example of a set of English patterns. Slot-names have their tag bolded, phrases are italicized, and d-tree items are underlined.

```
COCAINE-DEALER-ENG-PATS = (
  (TEACH (SL-NAME AGENT) (PHRASE TEACHES) (SL-NAME OBJECT) (PHRASE AT)
         (SL-NAME LOC) (PHRASE TO) (SL-NAME RECIP))
  (HUMAN (D-TREE DTR-1))
  (CHEM (PHRASE CHEMISTRY))
  (HS (SL-NAME REF) (PHRASE HIGH SCHOOL))
  (STATE (D-TREE DTR-2))
  (INFORM (SL-NAME AGENT) (PHRASE TOLD) (SL-NAME RECIP) (PHRASE THAT)
          (SL-NAME OBJECT))
  (CANCER (SL-NAME TYPE) (PHRASE CANCER))
  (INGEST (D-TREE DTR-3))
  (NOSE (PHRASE NOSE OF) (SL-NAME PARTOF))
  (DEF (PHRASE THE))
)
```

This execution of `(FR-TO-ENG COCAINE-DEALER-ENG-PATS A-CON* Q-CON*)` for some answer concept and some question concept will be as follows:

1. FR-TO-ENG sets *CUR-Q-CON to Q-CON*, *CUR-C-ANS to A-CON* (since it's the first call to FR-TO-ENG), and *CUR-FILLER to A-CON*.
2. FR-TO-ENG first looks at the top-level PREDicate of the answer concept A-CON*, and finds the corresponding pattern in the English patterns.
    a. If a pattern is found that matches the top-level PREDicate of A-CON*, then we follow the translation patterns as defined above, and glue the result of their translation together into the final return value.
    b. If a pattern is NOT found that matches the top-level PREDicate of A-CON*, then we simply return the name of the predicate exactly as it is.
3. Return the final translation result when there are no more sub-frames to translate based on the given input C-ANS.

[E] Example 1.1
Here is a simple example translation:

```
EX21-A-CON =
(TEACH AGENT (GEORGE)
       OBJECT (CHEM)
       LOC (HS REF (DEF))
       RECIP (STUDENTS))
Q-CON* = (QUESTION) ; Q-CON doesn't matter here, so we'll feed in
                    ; a dummy value, without loss of generality

(FR-TO-ENG COCAINE-DEALER-ENG-PATS EX21-A-CON Q-CON*)
; returns:
(GEORGE TEACHES CHEMISTRY AT THE HIGH SCHOOL TO STUDENTS)
```

Notice a few of the translation elements in the above example:

- When FR-TO-ENG is first called, we look for the TEACH predicate in our English patterns, and find it.
- In that pattern for TEACH, the first element is (**SL-NAME** AGENT), which is (GEORGE) in our EX21-A-CON. There is no matching predicate for GEORGE in our English patterns, so the literal phrase GEORGE is returned in that place.
- The (CHEM) predicate IS however found in our English patterns, and so we return the translation of CHEMISTRY in its place.

Notice, however, that a few of our translation patterns have decision trees that should decide their translation behaviors based on the status of our global variables. We will define another function, EVAL-D-TREE, that will give us the replacement patterns to use for each D-TREE in the English patterns.

Problem 2: (EVAL-D-TREE d-tree)

EVAL-D-TREE takes a single input, but has access to all 3 of the global variables that were set in Problem 1 above:

1. **D-TREE:** a *decision tree* that is passed in its entirety and *not* by the symbol name representing the D-TREE. For example, we would call EVAL-D-TREE by:
   `(EVAL-D-TREE D-TREE1)`
   and we would NOT call it by:
   `(EVAL-D-TREE 'D-TREE1)`

EVAL-D-TREE evaluates each AND- expression, and if any test with the AND-expression returns NIL, then EVAL-D-TREE evaluates the next AND-expression and so on, until an AND-expression is evaluated that returns a non-NIL value. At that point EVAL-D-TREE returns that leaf-pattern (which will be a leaf of the decision tree and should be a replacement pattern for FR-TO-ENG to use instead of the original pattern).

[!] Note: if *no* path in a D-TREE is satisfied, we will return NIL, indicating that no translation is to be made for the given pattern in FR-TO-ENG.

Here are 3 example decision trees:

```
DTR-1 = (
  (AND (GET-SF 'ROLE *CUR-FILLER)
       '((SL-NAME REF) (SL-NAME ROLE)))
  (AND (GET-SF 'TYPE *CUR-FILLER)
       '((SL-NAME TYPE)))
  (AND (GET-SF 'F-NAME *CUR-FILLER)
       '((SL-NAME F-NAME) (SL-NAME L-NAME)))
  (AND (GET-SF 'GENDER *CUR-FILLER)
       '((SL-NAME GENDER)))
)

DTR-2 = (
    (AND (SETQ AGNT (GET-NESTED-SF '(AGENT F-NAME) *CUR-FILLER))
         (SETQ GENDER (GET-NESTED-SF '(AGENT GENDER) *CUR-FILLER))
         (EQUAL AGNT (GET-NESTED-SF '(RECIP F-NAME) *CUR-C-ANS))
         (EQUAL GENDER '(MALE))
         '((PHRASE HE HAS) (SL-NAME OBJECT)))
    (AND '((PHRASE SHE HAS) (SL-NAME OBJECT)))
)

DTR-3 = (
    (AND (EQUAL (first *CUR-Q-CON) 'CAUSE)
         (IS-VARIABLE (GET-SF 'ANTE *CUR-Q-CON))
         (EQUAL (first (GET-SF 'TO *CUR-C-ANS)) 'NOSE)
         '((PHRASE BECAUSE) (SL-NAME AGENT) (PHRASE PUT)
           (SL-NAME OBJECT) (PHRASE INTO) (SL-NAME TO)))
)
```

[E] Example 2.1

```
*CUR-FILLER =
(HUMAN F-NAME (GEORGE)
       L-NAME (WHITE))

(EVAL-D-TREE DTR-1)
; returns:
((SL-NAME F-NAME) (SL-NAME L-NAME))
```

[E] Example 2.2

```
*CUR-FILLER =
(TEACH AGENT (HUMAN F-NAME (GEORGE)
                    GENDER (MALE))
       OBJECT (CHEM))

*CUR-C-ANS =
(PTRANS AGENT (DRUGEE)
        RECIP (HUMAN F-NAME (GEORGE))
        OBJECT (MONEY))

(EVAL-D-TREE DTR-2)
; returns:
((PHRASE HE HAS) (SL-NAME OBJECT))
```

Thus, your solution will use EVAL-D-TREE within FR-TO-ENG to translate frames, with the potential to remove ambiguities and make the translation more natural through inclusion of decision trees. We'll see some illustrations of this in the next section.

Additional Examples

Consider the following additional D-TREEs and amendments to the COCAINE-DEALER-ENG-PATS:

```
; [!] Add DTR-4:
DTR-4 '(
  (AND (EQUAL (GET-SF 'GENDER *CUR-FILLER) '(MALE))
       '((PHRASE HE)))
  (AND (EQUAL (GET-SF 'GENDER *CUR-FILLER) '(FEMALE))
       '((PHRASE SHE)))
  (AND '((PHRASE IT)))
)

; [!] Add the following patterns to COCAINE-DEALER-ENG-PATS:
(
  ...
  (PHYSICIAN (D-TREE DTR-4) (PHRASE IS) (SL-NAME OBJECT)
             (SL-NAME ROLE))
  (ATTENDED (D-TREE DTR-4) (PHRASE WAS) (SL-NAME AGENT)
            (SL-NAME OBJECT))
  ...
)
```

[E] Example 3.1

```
(FR-TO-ENG COCAINE-DEALER-ENG-PATS C-ANS-1 Q-CON1)
; returns:
(GEORGE TEACHES CHEMISTRY AT THE HIGH SCHOOL TO STUDENTS)
```

[E] Example 3.2

```
(FR-TO-ENG COCAINE-DEALER-ENG-PATS C-ANS-2 Q-CON2)
; returns:
(BECAUSE RICK PUT COCAINE INTO NOSE OF RICK)
```

[E] Example 3.3

```
(FR-TO-ENG COCAINE-DEALER-ENG-PATS C-ANS-3 Q-CON3)
; returns:
(THE ONCOLOGIST TOLD GEORGE THAT HE HAS TERMINAL CANCER)
```

Note the ambiguities that are resolved by inclusion of the D-TREEs in the above example: instead of "The oncologist told George that George has terminal cancer," the second "GEORGE" is replaced by the more natural pronoun "HE."

Here are several additional examples that illustrate the intended behavior of your assignment's functions, but also betray the difficulties implicit in constructing grammatically correct English sentences.

[E] Example 3.4

```
*CUR-FILLER = (HUMAN GENDER (MALE))

(EVAL-D-TREE DTR-4)
; returns:
((PHRASE HE))
```

[E] Example 3.5

```
(FR-TO-ENG COCAINE-DEALER-ENG-PATS
       '(PHYSICIAN F-NAME (DOC)
                   GENDER (MALE)
                   ROLE (ONCOLOGIST)
                   OBJECT (HUMAN F-NAME (GEORGE)))
       '(Q))
; returns:
(HE IS GEORGE ONCOLOGIST)
```

[E] Example 3.6

```
(FR-TO-ENG COCAINE-DEALER-ENG-PATS
       '(ATTENDED AGENT (HUMAN F-NAME (GEORGE))
                  OBJECT (HS))
       '(Q))
; returns:
(IT WAS GEORGE HIGH SCHOOL)
```

FAQ

- *Q: The spec says that each decision tree's leaf contains a "replacement" pattern if its path is satisfied. Is this replacement meant to actually modify the English patterns variable?*

  A: No, the replacement pattern is used only in the specific matched predicate pattern; i.e., the D-TREE will be re-evaluated for every subsequent time a predicate matches that pattern.

- *Q: Will there ever be multiple D-TREEs in a pattern or D-TREEs mingled with phrases and slot-names?*

  A: Yes, your solution should handle this.

- *Q: Can I assume that there is anything in the global variables before calling FR-TO-ENG?*

  A: It should not matter, as FR-TO-ENG sets these variables before doing anything else. You may, however, assume that the variables are at least defined.

- *Q: Will you test for the status of global variables? Should I be worried about what's in them before / after a test case is run for credit?*

  A: No, so long as you get the correct answer from FR-TO-ENG / EVAL-D-TREE. Do not, however, create any additional global variables for your solutions.

Assignment Specifications

- **Use the Provided Code Skeleton**
  A solution skeleton has been provided for you on CCLE. You should use this to structure your solutions, and may add any helper functions you see fit.

- **Use & Check the Piazza Forum!**
  If something is unclear, ask for clarifications; there may also be updates or fixes to the assignment that will be posted on the Piazza forum.

- **Using HW1, 2, 3 Solutions**
  For a uniform foundation in grading, it is assumed that you will be using the provided HW1, 2, 3 solutions should you decide to use any helper functions from the first three assignments.

- **Your Function Names Must Match These Exactly!**
  Failure to name your functions exactly as they appear in this homework will result in a grade of 0 for that problem!

- **Unfinished Functions**
  If you run out of time on a function, make sure you at least include that function definition and simply return 'UNIMPLEMENTED.

- **What to Submit**
  Submit one file, titled "*your-ID-number-here*.lsp" with all of your function definitions, and any helper functions you employed. DO NOT SUBMIT UNIT TESTS.

  E.g. I might submit a file named "555555555.lsp" (if 555555555 was my UID)

Grading

- **Problem Weights**
  All problems on this assignment carry equal weight; i.e., they are all worth the same amount of the final assignment grade.

- **Late Policy**
  Submissions that are *up to a day late* will receive a steep penalty as follows:
  - Before due date / time: 100% of score
  - 0 – 24 hours after due date / time: 70% of score
  - 24+ hours after due date / time: 0% of score