6.090 IAP '05 - Homework 7

Assigned Wednesday January 19th.
Due 10am Thursday January 20th.
Submit a print-out of your .scm file. Should you encounter difficulty printing, the file may be emailed to the 6.090 staff.

Thesaurus

"No, not the dinosaur" you explain to Ben Bitdiddle, "it allows you to look up related words." As usual, Ben Bitdiddle is trying to implement the wrong thing and it's your job to help him out.

Problem 1

A thesaurus is comprised of words and their synonyms. In order to simplify the implementation, we're going to write an abstraction for thesaurus entry, which contains a word and a list of its synonyms. Complete the constructors and selectors for the abstraction:

```
(define (make-entry word synonyms)
  to-be-completed)
(define (entry-word entry)
  to-be-completed)
(define (entry-synonyms entry)
  to-be-completed)
;; example usage:
(define e (make-entry "victory" (list "conquest" "triumph" "win")))
(entry-word e)
;Value: "victory"
(entry-synonyms e)
;Value: ("conquest" "triumph" "win")
```

Problem 2

A thesaurus is a list of entries. Of primary interest are two procedures: adding an entry to the thesaurus and looking a word up in the thesaurus. First, we'll implement adding entries:

```
(define empty-thesaurus nil)
```

```
(define (add-to-thesaurus entry thesaurus)
  to-be-completed)
(define t (add-to-thesaurus (make-entry "ball" (list "globe" "orb"
"rondure" "sphere"))
                            empty-thesaurus))
;Value: "t --> ((\"ball\" ...))"
(car t)
;Value: ("ball" ("globe" "orb" "rondure" "sphere"))
(cdr t)
;Value: #f
```

Once you've gotten a working `add-to-thesaurus`, download and evaluate hw7code.scm. (See supporting files in the Assignments section.) This will give you two thesauri: `simple-thesaurus` and `full-thesaurus`. The simple thesaurus is for simple testing, and the full version is for serious testing.

Problem 3

In order to look a word up in the thesaurus, we must search through the entries until we find one that matches. You should use `string=?` to compare strings. Remember to respect the entry abstraction. If the word is in the thesaurus, return the matching entry. If the word is not in the thesaurus, return false.

```
(define (lookup word thesaurus)
  to-be-completed)
(lookup "victory" simple-thesaurus)
;Value: ("victory" ("conquest" "triumph" "win"))
(lookup "defeat" simple-thesaurus)
;Value: #f
```

Problem 4

Supposing that you are given the procedure `pick-random`, which randomly picks an element of a list:

```
(define (pick-random lst)
  (if (null? lst)
      nil
      (list-ref lst (random (length lst)))))
```

Write the procedure `transmogrify`, which takes in a word and a thesaurus and returns one of the synonyms of the word, picked at random. If the word is not in the thesaurus, return the orginal word.

```
(define (transmogrify word thesaurus)
  to-be-completed)
(transmogrify "victory")
;Value: "triumph"
(transmogrify "defeat")
;Value: "defeat"
```

Problem 5

Now we can use this tool to rewrite sentences to make them sound more linguistically impressive. Assume a sentence is given as a list of words. You can use `string-explode` to convert a string into this form. To put the string back together, you can use `restore-string`. For example:

```
(string-explode "Assume a sentence is given as a list of words")
;Value: ("Assume" "a" "sentence" "is" "given" "as" "a" "list" "of"
"words")
(restore-string (list "this" "is" "a" "string"))
;Value: "this is a string"
```

Write the procedure `transmogrify-sentence` that takes in a sentence as a list of words and returns a new list of words with each of the words transmogrified.

```
(define (transmogrify-sentence sentence simple-thesaurus)
  to-be-completed)
(restore-string (transmogrify-sentence (string-explode "I went to the
store")
                                       simple-thesaurus))
;Value: "I went to the outlet"
(restore-string
 (transmogrify-sentence
  (string-explode "He hit the ball with the bat to end the game with
a grand victory")
  full-thesaurus))
;Value: "He strike the rundure with the bludgeon to end the game with
a impressive triumph"
```