**Introductory Scheme Exercise**

http://racket-lang.org and http://ranger.uta.edu/~weems/NOTES3302/NEWNOTES/NOTES02/notes02.rkt are assumed to be on your machine.

A simple game commences with two piles of stones. After agreeing on who-goes-first, each player in alternating turns may choose one of the following options:

 Take a stone from one of the piles.

 Take two stones, one from each of the two piles.

 *The loser is the player left with two empty piles and no option for taking stones.*

(Game-theoretic disclaimer: To have “perfect information”, the number of stones in each pile is known . . .)

1. Write the simplest possible Scheme code to indicate whether the first player has a win and the strategy for the first move, as found by an exhaustive search. The “luser” should type (play m n) where m and n are positive integers. The possible outputs are:

 "Win by taking from both"

 "Win by taking from first"

 "Win by taking from second"

 "Can't win"

 A few test cases:

(play 3 5)

(play 7 6)

(play 4 7)

(play 6 6)

(play 10 10)

2. The member? function in Notes 2 works for lats:

 > (member? 444 '(111 222 333 20))

 #f

 > (member? 111 '(111 222 333 20))

 #t

 Modify it so it also works for lists with arbitrary elements:

 > (member? '(444 333) '(111 (222 333) 20))

 #f

 > (member? '(111 222) '(555 (111 222) 333 20))

 #t

3. Write code to:

a. define a name associated with an empty list.

b. Insert a list or atom at the beginning of your named list (use set!)

4. The simple code for part 1. is horribly inefficient since it recomputes function values. Modify the code, using parts 2. and 3., to maintain a list of pairs for each case of “winners” and “losers”.

 After running a few cases, print your “caches” to reveal the secret of this game.