# CS 2500 Exam 1 HONORS Solutions – Fall 2013

**Problem 1** Design the function `even-dogs?` that takes a list of symbols and returns true of the symbol `'dog` occurs in the list an even number of times. 8 POINTS

```
;; even-dogs? : [Listof Symbol] -> Boolean
;; Does 'dog appear in los an even number of times?
(define (even-dogs? los)
  (cond [(empty? los) true]
        [else (if (symbol=? 'dog (first los))
                  (not (even-dogs? (rest los)))
                  (even-dogs? (rest los)))]))

(check-expect (even-dogs? '()) true)
(check-expect (even-dogs? '(cat dog)) false)
(check-expect (even-dogs? '(cat dog dog)) true)
(check-expect (even-dogs? '(cat dog elephant dog)) true)
```

**Problem 2** Design a function `shift-x` that given a list of `Posn`s and a number `n` (which may be negative or positive), shifts each posn in the list by `n` along the x-axis, *unless* the posn is the origin (0,0). 9 POINTS

You should design helper functions as needed, but they should be designed according to the recipe.

```
;; shift-x : [Listof Posn] Number -> [Listof Posn]
;; add n to the x field of each posn in lop
(define (shift-x lop n)
  (cond [(empty? lop) empty]
        [else (cons (shift-x-nonorigin (first lop) n)
                    (shift-x (rest lop) n))]))


;; shift-x-nonorigin : Posn Number -> Posn
;; shift p by n along x-axis, unless p is origin
(define (shift-x-nonorigin p n)
  (cond [(posn=? p (make-posn 0 0)) p]
        [else (make-posn (+ n (posn-x p)) (posn-y p))]))

;; posn=? : Posn Posn -> Boolean
;; Are the two posns equal?
(define (posn=? p1 p2)
  (and (= (posn-x p1) (posn-x p2))
       (= (posn-y p1) (posn-y p2))))

;; Examples/Tests
(check-expect (shift-x-nonorigin (make-posn 0 0) 5) (make-posn 0 0))
(check-expect (shift-x-nonorigin (make-posn 2 0) 5) (make-posn 7 0))

(check-expect (shift-x (list (make-posn 0 0) (make-posn 3 4)) -5)
              (list (make-posn 0 0) (make-posn -2 4)))
```

**Problem 3** The local meteorological society keeps a list of records about the weather each day. They track the following attributes: zip code, humidity (as a percentage), and high and low temperatures (in Fahrenheit) for the day. 12 POINTS

Here is the data definition for a weather record:

```
(define-struct weather (zip humidity hi lo))
; A Weather is a structure:
;    (make-weather String Number Number Number)
; interpretation: (make-weather z hum high low) is a
;   day's weather record where:
; - z is the 5-digit zip code where data was collected
; - hum is the humidity as a percentage
; - high and low represent the day's high and low
;   temperatures in degrees Fahrenheit, and high is
;   greater than or equal to low
```

The meteorological company has just been informed of a problem with temperature readings at all locations in zip code 02138. The high and low temperatures on file for this zip code are 4 degrees higher than the actual high and low temperatures of the day. Design a function `adjust-temps` that takes a list of weather records, a string representing the zip code, and a number `adjustment`, and produces a list of weather records that contains all the records in the input list but with the high and low temperatures in any record with the given zip code replaced by high+`adjustment` and low+`adjustment`, respectively.

Using your function, the meteorological society can fix its list of weather records for October 17th, called `lowr-oct-17-2013`, by running `(adjust-temps lowr-oct-3-2012 "02138" -4)`.

Again, design helper functions as needed, but they should be designed according to the recipe.

```
;; adjust-temps : LoWR String Number -> LoWR
;; Produce a list of weather records that is the same as given list
;; except that the high and low temps of all weather records for the
;; given zip code are adjusted.  Note: adjustment may be +ve or -ve.
(define (adjust-temps lowr zip adjustment)
  (cond [(empty? lowr) empty]
        [else (cons (adjust-if-zip (first lowr) zip adjustment)
                    (adjust-temps (rest lowr) zip adjustment))]))

;; adjust-if-zip : Weather String Number -> Weather
;; If weather record is for given zip code, then adjust high and
;; low temps.
(define (adjust-if-zip w zip adjustment)
  (cond [(string=? (weather-zip w) zip)
         (make-weather zip
                       (weather-humidity w)
                       (+ (weather-high w) adjustment)
                       (+ (weather-low w) adjustment))]
        [else w]))

;; Examples/Tests
(check-expect (adjust-if-zip w1 "02138" -4) w1)
(check-expect (adjust-if-zip w2 "02138" -4)
              (make-weather "02138" 60 66 46))
(check-expect (adjust-if-zip w2 "02138" 4)
              (make-weather "02138" 60 74 54))

(check-expect (adjust-temps lowr1 "02138" -4)
              (cons w1
                    (cons (make-weather "02138" 60 66 46)
                          (cons w3 empty))))
```

**Problem 4** Note the similarities and differences between `shift-x` from Problem 2 and `adjust-temps` from Problem 3. Design a function that abstracts over the differences and then use it to re-implement `shift-x` and `adjust-temps`.

13 POINTS

```
;; [Listof X] [X Y -> Z] Y -> [Listof Z]
(define (mymap l f n)
  (cond [(empty? l) empty]
        [else (cons (f (first l) n)
                    (mymap (rest l) f n))]))

;; [Listof Posn] Number -> [Listof Posn]
(define (shift-x1 lop n)
  (mymap lop shift-x-nonorigin n))

;; LoWR String Number -> LoWR
(define (adjust-temps1 lowr zip adjustment)
  (local ((define (adjust-if-this-zip w adjustmt)
            (adjust-if-zip w zip adjustmt)))
    (mymap lowr adjust-if-this-zip adjustment)))
```