CS2500 Exam 2 — Fall 2013

Your Name: _____

Instructor: _____

- This exam is open-book, open-notes. You may use any books, any notes, any written materials you brought along. Keep in mind that "design" does not mean "write down a function" or "write down what you know."
- **We will probably take off points for any notes that have no connection to the actual solution.** So erase or cross all out **all extraneous, disconnected** parts. For example, if the problem does not call for a template but you need one, put it into scratch space and cross it out once you are done.
- Your solutions may use all `ISL+lambda` syntax and functions. This means you *may* use the existing loop functions and you *may* use `lambda` to define simple functions (without contract or examples) in conjunction with such uses; you don't have to use either. One solution may refer to functions defined in others, but do provide a pointer.

  You may use the `List-of` notation to denote lists of specific data. You may interpret `Number` as `Real`.
- Make sure that you have six problems. Write down the answers in the space provided.
- **We will assign a score of 0 for: (1) any attempt to use electronic tools (laptops, phones, etc.), (2) any attempt to obtain solutions from someone else, or (3) for not following a proctor's instructions.**
- In case the staples come out of your completed test, write your name at the top of every page .

#	points	base
1		12
2		8
3		16
4		14
5		8
6		8
Total		66

Good luck!

Problem 1 (i) Identify the correct/incorrect data definitions below and explain in fewer than 15 words why they are correct/incorrect. 12 POINTS

1.
```
(define-struct snake (head tail))
;; A Snake is one of:
;; -- (make-snake Posn Snake)
```

2.
```
(define-struct 3tree (left middle right))
;; A TTree is one of:
;; -- Symbol
;; -- (make-3tree TTree TTree TTree)
```

Solution

(ii) Data definitions serve two roles: data construction and data recognition.

1. Construct one example per data definition:

```
(define-struct container (name content file))
;; A Container is a
;;     (make-container String [List-of Box] File).
;; A Box is one of:
;;    -- a Container
;;    -- a File
;; A File is a String.
```

2. Name or construct an instance of Fun:

```
;; Fun is a [String Number -> Number]
```

Solution

Problem 2 Develop templates for these data definitions:

8 POINTS

```
(define-struct leaf (val))
(define-struct straight (next))
(define-struct branch (left right))
;; A [Forest X] is one of:
;;   --  empty
;;   --  (cons [Tree X] [Forest X])
;;
;; A [Tree X] is one of:
;;   --  (make-leaf X)
;;   --  (make-straight [Tree X])
;;   --  (make-branch [Tree X] [Tree X])
```

Solution

(Page left intentionally blank.)

Problem 3 Design a program called rainfall that consumes a list of numbers representing daily rainfall amounts as entered by a user. The list may contain the number -999 indicating the end of the data of interest. Produce the average of the non-negative values in the list up to the first -999 (if it shows up).

16 POINTS

Solution

(Page left intentionally blank.)

Problem 4 Here is a data definition for lists that contains at least one item: 14 POINTS

```
;; [LOX1 X] is one of:
;; --  (cons X empty)
;; --  (cons X [LOX1 X])
```

(i) Design the function `join2`, which consumes two pieces of data: `l`, an instance of [`LOX1` X], and `x`, an `X`. It creates another list by inserting `x` between all pairs of neighboring elements in `l` (if there are any).

Solution

(ii) Design the function `join`, which consumes an arbitrary list `l` of `X`s and an instance of `X`. It inserts the latter between all pairs of neighboring elements in `l` (if there are any).

Solution

Problem 5 Design `zist`. The function consumes two lists of `Posns`. For each pair of corresponding `Posns` on the two lists, it computes the geometric distance. If there is a `Posn` on one list but no corresponding `Posn` on the other list, it computes the distance to the origin. 8 POINTS

The geometric distance between two `Posns` is computed as follows:

```
;; Posn Posn -> NonnegativeNumber
;; computes the distance between two points
(check-expect (distance (make-posn 1 1) (make-posn 4 5)) 5)
(define (distance p q)
  (sqrt
    (+ (sqr (- (posn-x p) (posn-x q)))
       (sqr (- (posn-y p) (posn-y q))))))
```

Solution

Problem 6 Inspect the following data definition:

8 POINTS

```
(define-struct leaf (val))
(define-struct fork (left right))
(define-struct straight (next))
;; An NTree is one of:
;;  --  (make-leaf Number)
;;  --  (make-fork NTree NTree)
;;  --  (make-straight NTree)
```

Design the function `split`. It consumes two pieces of data: `t`, an `NTree`, and `r`, a `Number`. It creates a new `NTree` by turning all `leaf`s in `t` into a `branch` with a `leaf` in each field:

- If `r` is smaller than the `val` field, `r` goes into the new left `leaf` and the `val` field becomes the right sub-tree.
- If `r` is greater than the `val` field, `r` goes into the new right `leaf` and the `val` field becomes the left sub-tree.

You may assume that `r` is not equal to any `Number` in `t`.

Solution

(Page left intentionally blank.)