CS2500 Exam 2 — Fall 2012

Name: _____

Student Id (last 4 digits): _____

Section (Shivers/Vona/Razzaq/Ahmed): _____

- Write down the answers in the space provided.
- You may use the usual primitives and expression forms, including those suggested in hints; for everything else, define it.
- The phrase "design this function/program" means that you should apply the design recipe. You are *not* required to provide a template unless the problem specifically asks for one. Be prepared, however, to struggle with the development of function bodies if you choose to skip the template step.
- To save time writing, you may write
  `(sqr 3)` $\rightarrow$ `9`
  instead of
  `(check-expect (sqr 3) 9)`
  You may also write the Greek letter λ instead of `lambda`, if you wish.
- Some basic test taking advice: Before you start answering any problems, read *every* problem, so your brain can be thinking about the harder problems in background while you knock off the easy ones.

Problem	Points	/out of
1		/ 8
2		/ 10
3		/ 11
4		/ 28
Extra		/ 7
Total		/ 64
Base	**57**	

Good luck!

Problem 1 Suppose we have the two lists

8 POINTS

```
(define a '(1 2))
(define b '((3 4) (5 6)))
```

What do each of the following expressions produce?

1. `(append a b)`
2. `(list a b)`
3. `(cons a b)`
4. `(apply append a b)`

Problem 2 Suppose we use the following data definition to represent binary trees with numbers at the leaves: 10 POINTS

```
;;; A BT is one of:
;;; - Number
;;; - (make-node BT BT)
(define-struct node (left right))
```

Here are two example binary trees:

```
(define tree1 (make-node (make-node 10 9)
                         (make-node 3
                                    (make-node 1 5))))
(define tree2 7)
```

Design a function to flatten a tree into the list of its elements, read left-to-right. So, `tree1` should flatten out to the list

```
(list 10 9 3 1 5)
```

Problem 3 Let's continue working with the binary trees of the previous problem. Two binary trees have the same *shape* if their structure matches, when we ignore the actual values at the leaves. So these this tree has the same shape as `tree1` from the previous problem:

11 POINTS

```
(define tree3 (make-node (make-node 9 10)
                         (make-node 5
                                    (make-node 0 -7))))
```

Design a function, `same-shape?` to determine if two trees have the same shape.

Problem 4 [Note: this problem has four parts. Make sure you answer all of them.] 28 POINTS

We can represent songs in a music collection with the following data definition:

```
;;; A Song is a (make-song String String Number)
(define-struct song title artist length)
;;; where the length of the song is given in seconds.

(define song1 (make-song "Hey, Jude" "The Beatles" 431))
(define songs (list song1
                    (make-song "U Smile" "Justin Bieber" 197)
                    (make-song "Free Bird" "Lynyrd Skynyrd" 608)))
```

1. Design a function, `total-time`, that consumes a list of songs and produces the total length of all the songs. Write your function using a loop function—but do not use `apply`.

2. Your instructors are *huge* Justin Bieber fans[1] We'd like a way to take a play list of songs and determine if the list contains any songs by Justin.

   Design a function, `any-bieber?` that returns true if the input list has one or more songs by artist `"Justin Bieber"`. Again, use a loop function; again, don't use `apply`.

3. We've been playing Justin Bieber[2] at the weekly CS2500 staff meeting. All semester. (We think it brightens up the mood.) To our dismay, not only have some of the tutors and TAs refused to join in when we close the meetings with our weekly sing-along of "As long as you love me," they've actually begun discussing writing a function that would take a list of songs and a string naming an artist, and remove all the songs in the list by the given artist. Perhaps you could help?

   Design a function, `remove-artist`, that takes a list of songs, and a string giving the name of an artist, and strips all the songs from the input list by that artist. As usual, use a loop function, and no `apply`.

[1] Razzaq prefers the rawer, grittier edge of his pre-*Misteltoe* work, before he sold out and went commercial, while Shivers maintains that *Believe* is some of his deepest and most harmonically complex work to date—but, really, pretty much everyone's a fan.

[2] Usually *My World 2.0*, but sometimes we'll switch things up with some *Believe* for variety.

4. Give a contract and purpose statement for the following function.

```
(define (fred xs)
  (quicksort xs
             (lambda (a b) (< (song-length a)
                              (song-length b)))))
```

[Here is some more space for the previous problem.]

Problem 5 (Extra credit) 7 POINTS

You can define binary trees that contain values at every node of the tree, not just at the leaves. For example, this data definition lets us represent binary trees that have a numeric grade in the range $[0, 100]$ at every node in the tree:

```
(define-struct node (left val right))

;;; A GradeTree is one of:
;;; - a Grade
;;; - (make-node GradeTree Grade GradeTree)
;;;
;;; A Grade is a number in the range [0,100].
```

Here are three example GradeTrees:

```
(define tree1 (make-node (make-node 3 0 1)
                         10
                         (make-node 8 100 (make-node 1 2 3))))
(define tree2 (make-node (make-node 1 0 10)
                         3
                         105))
(define tree3 (make-node (make-node 1 5 8)
                         10
                         (make-node (make-node 11 17 18)
                                    23
                                    87)))
```

A GradeTree node is *ordered* if

- all the numbers in its left child are smaller than the node's value,
- all the numbers in its right child are greater than the node's value, and
- the left and right subtrees are both ordered.

(Note that this is a recursive definition.) A tree *leaf* (being just a number) is considered to be trivially ordered.

So, for example, `tree1` is *not* ordered, because the left subtree `(make-node 3 0 1)` is not ordered. Likewise, `tree2` is also not ordered, because the left child contains the numbers $\{0, 1, 10\}$, but 10 isn't less than root node's value, 3.

On the other hand, `tree3` *is* ordered.

Ordered trees are useful. We can, for instance, check a large ordered tree to see if it contains a given number much more quickly than we can check an unordered tree, or the equivalent list of numbers.

Please design a function `tree-ordered?` that takes a GradeTree and determines if the tree is ordered or not. Note: for credit, your program must be simple, elegant and efficient: the amount of work it does should be proportional to the total number of numbers in the tree.

Hint: you may find it useful to declare a helper function...