University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Fall 2008 Instructor: Dan Garcia 2008-09-22

CS3L Quest

Personal Information

Last name	
First Name	
Last two letters of your login: `cs3-`	
Student ID Number	
The name of the TA for the lab ***you attend***	
Name of the person to your Left	
Name of the person to your Right	
All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS3L who have not taken it yet. ***(please sign)***	

Instructions

- Please turn off all cell phones.
  Remove all hats & headphones.
- **We will drop your lowest score** for questions 1 through 4. Question 0 is compulsory.
- You have one hour to complete this quest.
  It is open book and open notes, no computers.
- Partial credit will be given for incomplete / wrong answers, so please write down as much of the solution as you can**.**
- **Use** `true` **instead of** `#t` **, and** `false` **instead of** `#f`, since they are equivalent. Handwritten `#t` and `#f` unfortunately look too much alike.
- For these questions you only need the functions from the following sections (listed in the back page of the book): **Words and Sentences**, **Arithmetic**, **True and False** and **Variables**.
- Please comment on the exam below. Rate its difficulty (0 = cake, 5 = impossible), fairness (0 = unfair, 5 = fair), and **feel free to add any other comments that come to mind**.
- Difficulty (0=easy, 5=hard):
- Fairness (0=unfair, 5=fair):
- Other comments? (write here)

Grading Results

Question	*Max. Points*	*Points Earned*
0	**2**	
1	**6**	
2	**6**	
3	**6**	
4	**6**	
Subtotal	**26**	
Min (of 1-4)	**6**	
Total	**20**	

Comments:

Name: ______________________________

Question 0: "Say, what's the BIG idea?!" (2 pts, mandatory)

In one word, what is the "big idea" you'll learn in CS3L?
It is fundamental to *all* computer science & engineering.
(It allows you to drive a car without knowing how it works.)

□

Question 1: If it smells that good, it must be potpourri... (6 pts)

a) Fill in the blanks. If the expression returns an error, write ERROR and explain why:

`(last (first (butlast 'cheers)))` ➔
`(- (first '(10 9)) (bf '(8 7)))` ➔
`(or #f (and 'true (not 'false)) 'maybe not)` ➔

b) Add *only* quotes and parentheses to the following line to return the sentence shown.

```
sentence   bl   sentence   word   word   quote   s   bf        ➔        (words)
```

Question 2: Can you find debug? (6 pts)

We've tried to write `exactly-one?` that takes two Boolean inputs and returns `true` if (and only if) *exactly one* of its inputs is true. Unfortunately, we have a bug. Fill in the sentence; when writing a Boolean value, write `true` or `false`, not `#t` or `#f`, since those look remarkably alike.

```
  (define (exactly-one? a b)
1    (cond (a (not b))
2          ((and a b) true)
3          ((or  a b) true)
4          (else      true)))
```

a	b	(exactly-one? a b)
false	false	false
false	true	true
true	false	true
true	true	false

"Calling `(exactly-one? ______ ______)` *should* return ______ but instead returns ______.

Changing line # __ to ______________________ fixes the bug so it now works as advertised."

Question 3: This question is in the bag! (6 pts)

A `bag` is a new data type consisting of all its *items* (a sentence of objects) and its total *weight*. Your job is to finish writing the constructor `make-bag` and the selectors `bag-items` and `bag-weight`.

```
(define (make-bag items weight) ( ______________ 'bag-containing items 'weighs weight ) )

(define (bag-items bag)    ________________________________________________________ )

(define (bag-weight bag)   ________________________________________________________ )
```

Name: ____________________________________

Question 4: Difference Between Dates (6 pts)

You have already written `century-day-span` for the first mini-project. We would like you to write `iso-century-day-span` that **extends** `century-day-span` and the *Difference Between Dates* case study to handle dates between January 1st 1900 to December 31st 1999 in a modified format, we'll call *iso*, i.e., *(year month day)*. For example, January 3, 1990 would be written `'(90 january 3)`. You may assume you have already written the function `century-day-span`. **You may not change any definitions from the case study, you may only add to them.** To refresh your memory, the `century-day-span` procedure spec is re-printed below. Here are some example calls to `iso-century-day-span`:

```
(iso-century-day-span ‘(90 january  3) ‘(90 january  9)) ➔ 7
(iso-century-day-span ‘(89 march   30) ‘(90 february 2)) ➔ 310
(iso-century-day-span ‘(84 january  1) ‘(85 january  1)) ➔ 367
(iso-century-day-span ‘(1  january  1) ‘(5  january  1)) ➔ 1462
```

You only need to write `iso-century-day-span` & any helpers you'll need.

```
(define (iso-century-day-span iso-date1 iso-date2)
```

(Here is a copy of the homework description for `century-day-span`, in case that helps)

Write a procedure `century-day-span` that takes two dates as arguments, and returns the number of days between them, including the argument dates themselves. Assume that the first argument date is earlier than the second.

Each date is a three-word sentence representing a legal date between January 1, 1901 and December 31, 1999. The first word is a month name (one of january, february, ..., december). The second is an integer between 1 and the number of days in the specified month, inclusive. The third is an integer between 1 and 99, inclusive; it represents a year in the 20th century.

Your procedure must be able to deal with leap years. A leap year between 1901 and 1999 is any year that's divisible by 4; its February has 29 days rather than 28, and therefore it has 366 days rather than 365. E.g.,

```
(century-day-span '(january  3 90) '(january  9 90)) ➔ 7
(century-day-span '(march   30 89) '(february 2 90)) ➔ 310
(century-day-span '(january  1 84) '(january  1 85)) ➔ 367
(century-day-span '(january  1  1) '(january  1  5)) ➔ 1462
```