THE UNIVERSITY OF BRITISH COLUMBIA
CPSC 110: MIDTERM EXAMINATION
OCTOBER 5TH, 2010

Name: _______________________    Student #: _________________
CS Dept. ID #: __________

Signature: _____________________    Lab Section: _____     Lecture Section:_____

Important notes about this examination
1. You have 90 minutes to write this examination.
2. Follow the design recipes! You have been given a copy of the HtDDD and HtDF recipes. Use them.
3. Put away your books, notebooks, laptops, cell phones... everything but pens, pencils, erasers and this exam.
4. Good luck!

Rules Governing Formal Examinations
1. Each candidate must be prepared to produce, upon request, a UBCcard for identification.
2. Candidates are not permitted to ask questions of the invigilators, except in cases of supposed errors or ambiguities in examination questions.
3. No candidate shall be permitted to enter the examination room after the expiration of one-half hour from the scheduled starting time, or to leave during the first half hour of the examination.
4. Candidates suspected of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action:
   - having at the place of writing any books, papers or memoranda, calculators, computers, sound or image players/recorders/transmitters (including telephones), or other memory aid devices, other than those authorized by the examiners;
   - speaking or communicating with other candidates; and
   - purposely exposing written papers to the view of other candidates or imaging devices. The plea of accident or forgetfulness shall not be received.
5. Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.
6. Candidates must follow any additional examination rules or directions communicated by the instructor or invigilator.
Problem 1 - Mechanisms (20 points)

In this problem you will be given a number of expressions and asked to write the value of each. In some cases there will be some definitions that precede the expressions.

<table>
<thead>
<tr>
<th>What is the value of ((+ 2 6))</th>
<th>What is the value of ((* 3 (/ 4 2)))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>What is the value of ((&gt;= 9 10))</td>
<td>What is the value of ((\text{string-append } &quot;\text{Tom}&quot; &quot; \text{&quot;Riddle&quot;}))</td>
</tr>
<tr>
<td>((\text{if} (\text{string=}? &quot;a&quot; &quot;b&quot;) &quot;\text{dog}&quot; &quot;\text{cat}&quot;))</td>
<td>Given ((\text{define } A &quot;\text{foo}&quot;))</td>
</tr>
<tr>
<td></td>
<td>What is the value of: ((\text{string-length } A))</td>
</tr>
<tr>
<td>Given ((\text{define } (\text{cube } x) (* x x x)))</td>
<td>Given ((\text{define } (f1 a) (* a 2)))</td>
</tr>
<tr>
<td>what is the value of: ((\text{cube } 2))</td>
<td>(define ((f2 \text{x}) (* \text{x} 5)))</td>
</tr>
<tr>
<td></td>
<td>what is the value of: ((f1 \text{2}))</td>
</tr>
</tbody>
</table>
Given

(define-struct dims (w l h))

(define D1 (make-dims 3 4 5))

(define (vol d)
  (* (dims-w d) (dims-l d) (dims-h d)))

what is the value of:
(dims-l D1)

what is the value of:
(vol D1)
Problem 2 - Kinds of Data definitions (20 points)

In this problem you will be given five small fragments of a problem description. Each describes some information in a problem domain that must be represented using data in a program.

In each case circle the form of data definition you would design. Choose the ONE MOST SPECIFIC answer that makes sense for this information.

You may find it helpful in some cases to assume you are working on a typical world program, in which typical constants like WIDTH, HEIGHT, etc. have been defined. But you do not need to know the specific values of any of those constants.

You may make any other reasonable assumptions you wish, but if they are essential to the correctness of your answer write your assumption down.

(A) the state of a traffic light, which can be red, yellow or green

Atomic: itemization of intervals
Interval: compound (structures)
Enumeration: arbitrary-sized (lists)

(B) the x coordinate of a car moving across the screen

Atomic: itemization of intervals
Interval: compound (structures)
Enumeration: arbitrary-sized (lists)

(C) the plane has x and y coordinates

Atomic: itemization of intervals
Interval: compound (structures)
Enumeration: arbitrary-sized (lists)
(D) an integer score between 1 and 5 inclusive

<table>
<thead>
<tr>
<th>atomic</th>
<th>itemization of intervals</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval</td>
<td>compound (structures)</td>
</tr>
<tr>
<td>enumeration</td>
<td>arbitrary-sized (lists)</td>
</tr>
</tbody>
</table>

Circle ONE

(E) the names of all the Twitter users

<table>
<thead>
<tr>
<th>atomic</th>
<th>itemization of intervals</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval</td>
<td>compound (structures)</td>
</tr>
<tr>
<td>enumeration</td>
<td>arbitrary-sized (lists)</td>
</tr>
</tbody>
</table>

Circle ONE
Problem 3 - Designing Data Definitions (24 points)

In this problem you will design three complete data definitions. You should consider all of them to be part of SEPARATE programs.

For some of these questions you may find it helpful to assume you are working on a typical world program, in which typical constants like WIDTH, HEIGHT, etc. have been defined. You may make any other reasonable assumptions you wish, but if they are essential to the correctness of your answer write your assumption down.

This question will be graded on how well you follow the design recipe for data definitions. Remember to consult the How to Design Data Definitions supplement to this exam and write all the parts of a data definition it says are required.

(A)
a rating is an integer either between -10 and -1 inclusive, or between 1 and 10 inclusive
(B)
the height of the airplane

(C)
A song, with a name and a popularity score. The popularity score is an integer between 0 and 10 inclusive.
Problem 4 - Designing Functions (24 points)

In this problem you will be asked to design three separate functions. In each case you will be given one or more data definitions to work with. You should consider all of them to be part of SEPARATE programs. You may make any reasonable assumptions you wish, but if they are essential to the correctness of your answer write your assumption down.

This question will be graded on how well you follow the design recipe for functions. Be sure to show ALL the parts of the function the recipe calls for. Please write down the stub and label it. You do not need to write a separate copy of the template.

(A) Given this data definition:

```
;; Size is Number
;; interp. the length of each side of a square
(define S1 4)
(define S2 10)
;;
(define (fn-for-size s)
  (... s))
```

Design a function called area that consumes Size and produces a number that is the area of the square. Remember that the area of a square is simply the length of a side multiplied by itself.
(B) Given this data definition:

(define-struct person (fn ln))
;; Person is (make-person String String)
;; interp. a person's first and last names
(define P1 (make-person "Harry" "Potter"))
(define P2 (make-person "Tom" "Riddle"))
#
(define (fn-for-person p)
  (... (person-fn p)
       (person-ln p)))

Please design a function named last-first which consumes a Person and produces a string with the last name followed by a comma, a space and the first name. So for example:

(last-first P1) would produce "Potter, Harry"
(C) Given this data definition:

`; `ListOfPrice is one of:
`; - empty
`; - (cons Number ListOfPrice)
`; interp. the individual prices of all items in a catalogue
(define LOP1 empty)
(define LOP2 (cons 32 (cons 12 (cons 45 empty))))
#
(define (fn-for-lop lop)
  (cond [(empty? lop) (...)]
    [else
      (... (first lop)
           (fn-for-lop (rest lop)))]))

Design a function called \texttt{markup} that consumes a \texttt{Number} and a \texttt{ListOfPrice} and produces a new \texttt{ListOfPrice} in which each price is the product of the original price and the number. For example:

\begin{verbatim}
(mark-up 1.10 (cons 20 empty)) would produce (cons 22 empty)
\end{verbatim}
Problem 5 - Completing a World Program (12 Points)

In this problem you will complete a partially complete world program. First read the description of the program's behavior below. Then read through the program, completing it in the marked places. This problem will be graded primarily on your proper handling of references from one data definition to another, in both templates and final code.

HINT: Do not get overwhelmed by this question. Do not try to absorb every detail. Instead skim the whole program, then consider the focused tasks we are asking you to do. Then go back through to gather the details you need to do that task.

The program shows an animation of a fireworks show, involving any number of fireworks, each of which has an x-position, y-position and a colour.

As time goes by each firework moves up the screen and the colour of each firework cycles from red to green to blue and back to red. The x-position does not change. The fireworks just keep going up and changing colors, they never explode.

;;; A Fireworks Show

(require 2htdp/image)
(require 2htdp/universe)

;;; Constants:
(define WIDTH 200)
(define HEIGHT 400)
(define SPEED 4) ; rate at which y decreases per tick

(define MTS (empty-scene WIDTH HEIGHT))

;;; Data definitions:

;;; Colour is one of:
;;; - "red"
;;; - "green"
;;; - "blue"
;;; interp. the color of a firework

Write the template for Colour here.
(define-struct fw (x y colour))
;; Firework is (make-fw Natural Natural Colour)
;; interp. the x and y coordinates and colour of a firework
(define FW1 (make-fw 10 HEIGHT "red"))
(define FW2 (make-fw 20 (* HEIGHT .8) "green"))
(define FW3 (make-fw 20 (* HEIGHT .9) "blue"))

Write the template for Firework here.

;; ListOfFirework is one of:
;; - empty
;; - (cons Firework ListOfFirework)
;; interp. a list of fireworks in the show
(define LOFW1 empty)
(define LOFW2 (cons FW1 (cons FW2 (cons FW3 empty))))

Write the template for ListOfFirework here.
;; Functions:

;; ListOfFirework -> ListOfFireWork
;; start the show with a call like (main LOFW2)
(define (main lof)
  (big-bang lof
    (on-tick tick-lofw) ; ListOfFirework -> ListOfFirework
    (to-draw render-lofw))) ; ListOfFirework -> Image

;; ListOfFirework -> ListOfFirework
;; produce list of results of ticking each fw in lofw
(check-expect (tick-lofw (cons (make-fw 10 400 "red")
    (cons (make-fw 20 430 "blue")
      empty)))
  (cons (make-fw 10 (- 400 SPEED) "green")
    (cons (make-fw 20 (- 430 SPEED) "red")
      empty)))

(define (tick-lofw lofw) ; this is the stub
  (cons (make-fw 10 20 "red") empty))

Write the complete code for tick-lofw here (and on the next page). If you follow the rules discussed last week you should end up with 2 helper functions in addition to tick-lofw. For the two helper functions you should write the signature, purpose and code. You do not need to write examples for those functions, but you may find it useful to do so.