NAME:

CS 1101
Exam 1
A-Term 2013

Question 1:	_____	(15)
Question 2:	_____	(30)
Question 3:	_____	(25)
Question 4:	_____	(30)
TOTAL:	_____	(100)

You have 50 minutes to complete this exam. You do not need to show templates (unless a problem states otherwise), but you may receive partial credit if you do. You also do not need to show test cases or examples of data definitions (unless a problem states otherwise), but you may develop them if they will help you write the programs.

Your programs may contain only the following Racket constructs:

define define-struct cond else if

empty? cons? cons first rest

`+ - * / = < > <= >=`

string=? string-length string-append image-width image-height
and or not

and the operators introduced by **define-struct**.

You may, of course, use whatever constants are necessary (*empty, true, false,* 0, etc.)

1. (**Function evaluation: 15 points**)
   Assume that these two functions have been defined:

```
;; triple:  Number -> Number
;; consumes a number and produces the number, tripled
(define (triple num)
  (* num 3))

;; difference:  Number Number -> Number
;; consumes two numbers and produces the result of subtracting the
;; smaller number from the larger number
(define (difference a b)
  (if (> a b)
      (- a b)
      (- b a)))
```

   Evaluate the following expression step by step, the way Racket would evaluate it. Show every step.

```
(difference 12 (triple (* 2 5)))
```

2. (**Test cases: 30 points**)
A programmer defines the following data definitions and signature/purpose:

```
(define-struct person (name age))
;; Person is a (make-person String Natural)
;;  interp:  (make-person name age) represents a person with
;;           name as the person's name
;;           age is the person's age (in years)

;; ListOfPerson is one of
;;   empty
;;   (cons Person ListOfPerson)

;; children:  ListOfPerson -> ListOfPerson
;; consumes a list of persons, and produces a list of only those persons from
;; the original list who are younger than 18 years old
```

Using `check-expect`, write a set of test cases to adequately test the function `children`.
Do not write the function definition. Just write the test cases.

3. (**Itemizations: 25 points**)
A college uses these data definitions for programs that manage different kinds of students:

```
(define-struct ugrad (id yog))
;; Ugrad is a (make-ugrad Number Number)
;; interp:  (make-ugrad id yog) is an undergraduate student with
;;          id as the student id
;;          yog is the student's expected year of graduation

(define-struct grad (id yog ta?))
;; Grad is a (make-grad Number Number Boolean)
;; interp: (make-grad id yog ta?) is a graduate student with
;;         id as the student id
;;         yog is the student's expected year of graduation
;;         ta? is true if the grad student is a teaching assistant

(define-struct contEd (id credits))
;; ContEd is a (make-contEd Number Number)
;; interp:  (make-contEd id credits) is a continuing ed student with
;;          id as the student id
;;          credits as the number of credits the student is currently taking
```

(a) (5 points) Provide a data definition for a new data type called `Student`, which encompasses the three kinds of students defined above.

(b) (10 points) Provide a template for functions that operate on `Student`.

(c) (10 points) Here is a signature and purpose for a function. Complete the function by writing the function definition.

```
;; defer-graduation:  Student -> Student
;; Consumes a student.  If the student is a continuing ed student,
;; the student is returned unchanged.  Otherwise, a new student is
;; produced that is the same as the original except that
;; the new student's year of graduation has increased by one year
```

4. (**Helpers and list functions: 30 points**)

(a) (15 points) Write a function that satisfies the following signature and purpose:

```
;; is-big?:  Image -> Boolean
;; consumes an image and produces true if the area of the image exceeds 500
```

(Hint: remember that Racket has two built-in functions, `image-width` and `image-height`, both of which consume an image, and return the width and height (respectively) of the image.)

(b) (15 points) Assume that `ListOfImage` is defined like this:

```
;; ListOfImage is one of
;;    empty
;;    (cons Image ListOfImage)
```

Write a function that satisfies the following signature and purpose. Your definition for `count-big-images` must call the function `is-big?` (from part (a)) as a helper function.

```
;; count-big-images:  ListOfImage -> Natural
;; consumes a list of images and counts the number of images in the list
;; that have an area exceeding 500
```

(end of exam)