

NAME:

**CS 1101**  
**Exam 2**  
A-Term 2010  
9am Lecture Section

Question 1: \_\_\_\_\_ (30)

Question 2: \_\_\_\_\_ (35)

Question 3: \_\_\_\_\_ (35)

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.

**You must provide a contract/purpose for each helper function you define.**

Your programs may contain only the following DrRacket constructs:

**define define-struct cond else**

and the following primitive operators:

*empty? cons? cons first rest list append*

+ - \* / = < > <= >=

*string=? string-length symbol=?*

**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. The International Olympic Committee hires you to write software that keeps track of Olympic athletes. The information kept about each athlete consists of the athlete's name, age, gender, country, and medals earned. The information about medals consists of the number of gold, silver, and bronze medals, and is represented by the following data definition:

```
;; medals is a struct
;; (make-medals number number number)
(define-struct medals (gold silver bronze))
```

- (a) (15 points) Write any additional data definitions needed to represent a list of athletes.

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

```
;; all-gold: list-of-athlete -> list-of-athlete
;; consumes a list of athletes and produces a list of only those
;; athletes who have earned at least one medal and for whom all medals earned
;; are gold medals
```

2. An airline keeps track of the destinations for non-stop flights from each city serviced by the airline. For example, for one particular airline, from Boston there are flights to New York and Baltimore. From New York there are flights to Atlanta and Dallas. From Baltimore there is a flight to Miami. From Atlanta there are flights to Detroit, Minneapolis, and Phoenix. There are no other flights on the airline's schedule.

Here is an example of DrRacket data that represents the flight schedule described above:

```
(define FROM-BOSTON
  (make-flight "Boston"
    (list
      (make-flight "New York"
        (list
          (make-flight "Atlanta"
            (list
              (make-flight "Detroit" empty)
              (make-flight "Minneapolis" empty)
              (make-flight "Phoenix" empty))))
          (make-flight "Dallas" empty)))
      (make-flight "Baltimore"
        (list
          (make-flight "Miami" empty))))))
```

For purposes of simplicity, you may assume that no cycles exist in flight schedules (for example, you would not have a schedule where there is a flight from Boston to New York, from New York to Philadelphia, and from Philadelphia back to Boston).

- (a) (10 points) Provide a set of data definitions from which the given example of data could be constructed.

(b) (25 points) Write a program that satisfies the following contract and purpose:

```
;; path-exists?: flight string -> boolean
;; consumes a flight and the name of a destination city and produces true if
;; there exists a way to fly from the departure point of the given
;; flight to the given destination
```

To help you understand what the function does, here is a set of test cases for the function:

```
(check-expect (flight-exists? FROM-BOSTON "Los Angeles") false)
(check-expect (flight-exists? FROM-BOSTON "Dallas") true)
```

```
;; in the following test case, notice that the function returns true when th
;; city you're departing from is the same as the city you specify as
;; the destination
```

```
(check-expect (flight-exists? FROM-BOSTON "Boston") true)
```

3. An astronaut travels to a planet and discovers a new life-form, which the astronaut calls a `grok`. Groks come in many different colors. Groks can reproduce only once, and when a grok reproduces, it has exactly 2 offspring. The following DrRacket data definitions are used to represent information about a grok's reproductive tree:

```
;; an info is a struct
;; (make-info string string)
(define-struct info (name color))

;; a gtree (grok tree) is either
;; 'stop
;; (where 'stop is used to indicate a grok has no offspring), or
;; (make-grok info gtree gtree)
(define-struct grok (data baby1 baby2))
```

- (a) (10 points) Provide an example of a gtree that represents a green grok named Henry who has two offspring, both of whom are red. The names of the two offspring are Dan and Don. Correct punctuation is required for full credit for this problem.

(b) (15 points) Provide templates for *all* data definitions given in this problem.

(c) (10 points) Write a program that satisfies the following contract and purpose:

```
;; count-blue: gtree -> number
```

```
;; consumes a gtree and produces the number of blue groks in the tree
```