THIRD QUIZ

You have 20 minutes from the start of class to complete this quiz. The usual instructions apply. Don't reinvent the wheel; for full credit, use functions from earlier problems where helpful in a later one. Good luck!

You may show lists in any of three ways: `(list 'AC 'DC)`, `'(AC DC)`, or `(cons 'AC (cons 'DC empty))`.

Pay close attention to the type of data each function expects as inputs and returns as output.

Problem 1 (4 points)

Evaluate each of the following expressions. Use this definition independently for each of the five parts:

```
(define L (cons 'Candle (cons 'Cake (cons 'Present (cons 'Card empty)))))
```

(a) `(first L)`

(b) `(rest L)`

(c) `(rest (rest L))`

(d) `(first (rest (rest L)))`

(e)
```
(cond
   ((empty? L) 'Happy)
   ((empty? (rest (rest (rest L)))) 'Birthday)
   (else 'ManyHappyReturns))
```

Problem 2 (4 points)

Complete the definition of `add-up-positive-items` below

```
;; add-up-positive-items:  list-of-number -> number
;; Return the sum of all the positive items on the list, ignoring negative items
(define add-up-positive-items
  (lambda (L)
    (cond
      ((_____________________________ L) _____________________________)

      ((_____________________________ 0) (+ (first L) (_____________________________)))

      (else (_____________________________)))))
```

Problem 3 (22 points)

The Irvine Animal Preserve keeps track of its animals using a structure:

```
(define-struct animal (name species yearborn weight))
```

where the name and species are strings and the year born and weight are numbers.

(a) (8 points) Complete the definition of this function according to the contract and purpose given. Examples and tests are not required for credit on this quiz, but it's still a good idea to think about them.

```
;; animal-lookup: string list-of-animal -> animal or empty
;; Given a string, return the animal on the list whose name matches the string.
;; If the list is empty or if no animal on the list has that name, return empty.
(define animal-lookup
  (lambda (n L)
```

(b) (6 points) Complete the definition of this function according to the contract and purpose given. (Hints: Think about what you need to return and how to get it and note that a name isn't an animal.)

```
;; animal-heavier?: string string list-of-animals -> boolean
;; The strings are animal names; return true if the first named animal weighs
;; more than the second, and false otherwise.
(define animal-heavier?
  (lambda (A1 A2)
```

(c) (8 points) Complete the definition of this function according to the contract and purpose given.

```
;; keep-older-animals: number list-of-animals -> list-of-animals
;; The number represents a year; return a list of just those animals on the
;; input list that were born in the specified year or earlier.
(define keep-older-animals
  (lambda (year L)
```