Name:_____

Question 0 (20 points)	
Question 1 (15 points)	
Question 2 (20 points)	
Question 3 (25 points)	
Question 4 (20 points)	
Total (100 points)	

You have 75 minutes to complete the exam. You may have with you a single piece of paper with notes on both sides. You may have *only* your sheet of notes, a single writing implement, and the exam on your desk (e.g., no cell phones, laptops, etc.).

You do not need to ases unless specifically asked. can not provide a complete solution.

While you wait, limber up your writing implement ational picture.

Question 0: Terminology (20 points)

```
;; AppleColor is "red", "green", or "yellow"
```
E

```
;; CrispNumber is a number between [1, 10]
```
F

```
(define-struct apple (weight color crunchiness))
```
I

```
;; make-apple: Number AppleColor CrispNumber -> Apple
;; Interpretation:
;;  -- weight is the weight of the apple in ounces
;;  -- color is the color of the apple
;;  -- crunchiness is how crisp the apple is
(define APPLE1 (make-apple 8 "red" 9))
#;
(define (apple-fun anApple)
  ... (apple-weight anApple) ...      ; Number
  ... (apple-color anApple) ...       ; AppleColor
  ... (apple-crunchiness anApple) ... ; CrispNumber
)

(define-struct banana (w
;; make-banana: Positive
;; Interpretation:
;;  -- weight is the weight of the banana in ounce
;;  -- ripe? is true if the banana is ripe, false
(define BANANA1 (make-banana 4 true))
#;
(define (banana-fun aBanana)
  ... (banana-weight aBanana) ...
  ... (banana-ripe? aBanana) ...)

;; a Fruit is either an Apple or a Banana
#;
(define (fruit-fun aFruit)
  (cond [(apple? aFruit) (apple-fun aFruit)]
        [(banana? aFruit) (banana-fun aFruit)]))

;; edible?: Fruit -> Boolean
;; consumes: a fruit
;; produces: true if the given fruit is edible, false otherwise
(define (edible? aFruit)
  (cond [(apple? aFruit) (apple-edible? aFruit)]
        [(banana? aFruit) (banana-ripe? aFruit)]))

(check-expect (edible? APPLE1) true)
```
K J O D E G M H A B

Match the term to the letter. Some terms may not appear (choose "Z") or may appear more than once (choose any letter).

_____ comment

_____ interval

_____ enumeration

_____ itemization

_____ parameter

_____ argument

_____ template

_____ clause

_____ constant

_____ unit test

____ Fruit

_____ predicate

_____ constructor

_____ selector

_____ structure definition

_____ constant definition

_____ function definition

_____ signature

_____ function call

_____ boolean

Question 1: Simple Function (15 points)

Design a function `rph->mpg` that converts rods per hogshead (rph) to miles per gallon (mpg). Show the signature, purpose statement, test(s), and definition for the function.

Domain specific information:

- 1 hogshead = 63 gallons
- 1 rod = 0.003125 miles
- 40 rph is approximately 0.001984126 mpg

  0.001984126 mpg = ((40 rph * 0.003125 miles) / 63 gallons)

Question 2: Cond (20 points)

The playing cards suits are spades, hearts, diamonds, and clubs. Design a function that *converts* suites to their corresponding images (i.e., it consumes a suit and produces the corresponding image—spades: ♠, hearts: ♥, diamonds: ♦ and clubs: ♣). Remember that DrRacket supports image literals. Don't try to try to create the images using image operations (e.g., circle, overlay, etc.). Just draw them.

Question 3: Data Definitions and Templates (25 points)

The Dunder Mifflin paper company sells paper in two sizes: Letter and A4 and three colors: white, yellow, and green. All paper orders are handled in a single unified interface. An order for letter paper needs to indicate the number of sheets, the paper color, and whether the sheets are lined. An order for A4 paper needs to indicate the number of sheets, the paper color, and whether the sheets have holes.

Develop a data definition for a PaperOrder and either a LetterOrder or an A4Order. Provide additional data definitions for other types, as necessary.

Question 4: Structures (20 points)

Consider the following definitions:

```
(define-struct date (day month year))
;; make-date: Number Number Number -> Date
;; interp: day    is the day of the date
;;         month  is the month of the date
;;         year   is the year of the date

(define-struct post (date content))
;; make-post: Date String -> Post
;; interp: date is the date of the post
;;         content is the content of the post
```

Design a function that accepts a post and a year and produces true if the given post was made during the given year (i.e., the year of the post's date is the same as the given year) and false otherwise.