CS135 Tutorial 03

Review of Structs

The data definition of structs is

```
(define-struct rect (top left width height colour))
;; A Rect is a (make-rect Num Num Num Num Sym)
;; Requires: width and height are non-negative
```

From this definition racket provides us with the following functions:

- Constructor: `(make-rect …)`
- Selectors: `(rect-top r)`, `(rect-left r)`, …
- Predicate: `(rect? r)`

Nutritional Facts

Nearly all packaged foods in Canada are required to display a nutritional facts table. We use a simplified version of that for this question.

```
(define-struct nutri-fact (name serving fat carbs protein))
;; A Nutri-Fact is a (make-nutri-fact Str Num Num Num Num)
;; requires: 0 < serving
;; fat, protein, carbs >= 0
;; fat + carbs + protein <= serving
```

All of the numerical fields are measured in grams and represent the number of grams of each macro-nutrient per serving.

Nutritional Facts Cont.

Here are a few nutri-fact examples:

```
(define coke-zero (make-nutri-fact "Coke Zero" 355 0 0 0))
(define cheerios (make-nutri-fact "Honey Nut Cheerios" 29 1.5 23 2))
(define cashews (make-nutri-fact "Cashews" 30 14 9 5))
(define ketchup (make-nutri-fact "Ketchup" 15 0 5 0.3))
(define tuna (make-nutri-fact "Canned Tuna" 55 1 0 11))
```

Build a template for a function that consumes a `nutri-fact`

Nutritional Facts Cont.

Now that you've built the template, use the template to build the following functions:

1. `calories`: consumes a `nutri-fact` and calculates the total calories per serving. Each gram of fat is 9 calories, and each gram of protein or carbs is 4 calories.
2. `valid-nutri-fact?`: consumes a `nutri-fact` and produces `true` for a valid `nutri-fact` and `false` otherwise.