CMSC 15100 Introduction to Computer Science I
Homework 4

Important: Please add to (require "../include/cs151-core.rkt") at top of your file. Your answers should be saved in a single .rkt file and submitted via Canvas before the start of the next lecture. At the top of the file for this assignment and all future assignments include a comment,

```
;; Homework 4
;;
;; your name
;; your CNET id
```

If your code doesn't compile, it may not be graded. Please double check to make sure your code runs before submitting!

Problems

1. (7 points) The RaCalendar (short for Racket Calendar) is a fictional calendar app that keeps track of the activities that need to be done this week. RaCalendar maintains a list of `Activitys`, each of which has a `Description`, a `Day-of-Week` for which day of the week (Monday through Sunday) the `Activity` will occur on, and a `Location`.

   (a) Create types `Description`, `Day-of-Week`, `Location`, `Activity`, and `Calendar` representing the objects that are used in the RaCalendar. Before each `define-type` or `define-struct`, write a one-line comment justifying why you chose the type you did. (Hint: Activity will be a structure. Calendar will be a list of Activity)

   (b) Make `Activity` variables for the four scheduled parts of CS 151 that take place each week (three lectures and a lab), then create a `Calendar` with these `Activitys` in it.

   (c) Write a function `weekend?` that takes as input an `Activity` and says if the activity occurs on the weekend. Your function should use `match` and not `cond` or `if`.

2. (10 points) When a pine tree grows up, it sometimes releases a pine cone. On the other hand, pine cones sometimes grow into sturdy pine trees, but sometimes the conditions aren't right and they lie dormant. These relationships are captured by the following two type definitions which represent a "family tree" for pine trees:

```
(define-struct Pine-Tree
  ([seed : (U 'nothing Pine-Cone)]
   [height : Real]))

(define-struct Pine-Cone
  ([grows-into : (U 'nothing Pine-Tree)]
   [number-of-bristles : Integer]))
```

   (a) Make a function `seed-bristles` that takes as input a `Pine-Tree` and tells you how many bristles its seed has. If the input `Pine-Tree` has no seed, the result should be 0. This function has type `Pine-Tree -> Integer`.

(b) Make a function `germinate` that takes a `Pine-Tree` and adds a new seed `Pine-Cone` to the pine tree. For this new seed, the default number of bristles should be 40, and by default the pine cone should grow into nothing. The type of this function should be `Pine-Tree -> Pine-Tree`.

If the input `Pine-Tree` already has a `Pine-Cone` seed, you should just return the input `Pine-Tree` unmodified.

(c) It's important to know if a tree has a heritage of tallness. Make two mutually recursive functions, `max-tree-descendant-height` and `max-cone-descendant-height`, that compute the maximum height of a tree in the descendants of the input pine tree or cone, respectively.

3. (8 points) This question deals with the definition of the natural numbers as type `Nat` from class.

(a) Implement the greater-than-or-equal-to operator. More specifically, you should write a function with type annotation `(: my-gte : Nat Nat -> Boolean)` that checks if the first input represents a natural number that is greater than or equal to the second input.

(b) Implement the function `my-odd?` on `Nat` that checks if a `Nat` represents an odd number. The type of `my-odd?` is `Nat -> Boolean`.

(c) (1 point extra credit) Implement the `my-quotient` function on `Nat`. The type of the function should be `Nat Nat -> Nat`.