CMSC 15100 Introduction to Computer Science I

Homework 5

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 5
;;
;; 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) For this problem, copy the definitions of the types for RaCalendar from your HW4 file. If it isn't already, make sure that the type of `Calendar` is

   ```
   (define-type Calendar (Listof Activity))
   ```

   Also copy over the variables for CS151. This calendar should be used for your tests. When you want to test a function whose output type is `Calendar`, you'll have to define a new variable of type `Calendar` for the expected output.

   For the tasks in this problem, you should use `map` and `filter` to manipulate the activities in a calendar. Do not write any recursive functions. You can use as many helper functions as you like, however.

   (a) Since weekend activities are just for fun, make a function that keeps the weekday activities in a calendar, and gets rid of all of the weekend activities. As an example of what we're looking for, the basic outline of this function will look like

   ```
   (: weekday-activities : Calendar -> Calendar)
   (define (weekday-activities cal)
     (filter weekday? cal))
   ```

   (b) Make a function `back-one` that moves all `Activitys` in a `Calendar` backwards in the week by one day.

   (c) Make a function `ry251` that counts how many `Activitys` in a `Calendar` are taking place in Ryerson 251.

   Hint: in addition to the functions mentioned above, the `length` function will be useful

2. (6 points) Just write down type annotations for the following functions; you don't need to implement these functions. In order to avoid Racket errors about no function definition, either comment out your type annotations or add a definition that raises an error such as "not yet implemented". If you choose to comment out, make sure your type annotation is valid Racket code!

(a) A function `apply-one-or-the-other` that takes three arguments: the first input is a function from an arbitrary type to `Real`, the secoond input is a function from a different arbitrary type to `Real`, and the third input is a member of a union type of those two arbitrary types. The result of the function is to apply the correct function to the third input depending on which part of the union type the third input comes from.

(b) A function `find-maximizer` that takes in a list with arbitrary type and a function from that type to `Real`, and outputs the element of the list that maximizes the function. For example, if the list is a `(Listof String)` and the function is `string-length`, then `find-maximizer` will output the longest string in the list.

(c) A function `combine-lists` that takes in two lists of the same type and also a "combining function" that is applied to the first elements of the two lists, the second elements of the lists, the third elements of the lists, etc, to produce a new list. The output list is allowed to have a different type than the inputs.