CS 1101 - A-term 16

Homework 6 - Higher-order functions; accumulators; mutable variables

Due: Tuesday, October 11 at 5pm

Read the expectations on homework.

Assignment Goals

- To be able to use higher-order function to simplify the writing of list problems
- To make sure you can create linked data structures and write programs using mutable variables
- To make sure you can write programs in accumulator-style

The Assignment

There are two parts to this assignment. In Part 1, you will use map and filter to re-write some of the list functions we developed a couple of weeks ago. In Part 2, you'll implement a simple email system.

Part 1 - Rewriting HW3 problems using filter/map

In Homework 3 you wrote several functions that operated on a list of television ads. Use the following data definitions from Homework 3 for problems 1-4:

```
(define-struct ad (political? name duration production-cost national? time-of-day repetitions))
;; an Ad is a (make-ad Boolean String Natural Natural Boolean String Natural)
;; interp:  a television ad, where
;;          political? is true if the ad is a political ad, false otherwise (for product ad)
;;          name is the name of the politician or the product the ad is for
;;          duration is the length of the ad (in seconds)
;;          production-cost is the cost to produce the ad (in thousands of dollars)
;;          national? is true if the ad is to be aired nationally (false if locally)
;;          time-of-day is one of "P" for primetime, "D" for daytime, "O" for off-hour
;;          repetitions is the number of times the ad is to be played

;; a ListOfAd is one of
;;   empty
;;   (cons Ad ListOfAd)
```

Make sure you name your functions exactly the same as the names given in the problems below.

1. Using filter and/or map, rewrite the function `count-political-ads` (from Homework 3) that consumes a list of ads and produces the number of ads in the list that are classified as political ads. You may use the built-in Racket function `length`, which consumes a list and produces the number of items in the list.

2. Using filter and/or map, rewrite the function `primetime-ads` that consumes a list of ads and produces a list of all the ads airing in primetime.

3. Using filter and/or map, rewrite the function `politicians-sponsoring-ads` that consumes a list of ads and produces a list of strings. The list that is produced contains the names of the politicians who have political ads (it's OK if the resulting list contains duplicate names).

4. Using filter and/or map, write a new function `cheap-to-produce` that consumes a list of ads and a number and produces a list of ads. The list that's produced contains those ads for which the production costs are less than the given amount. (The given amount is in thousands of dollars.)

Part 2 - A Simple Email System

In a simple email system, users have usernames and a collection of messages in their mailbox. A message consists of the ~~sender's username~~ user who sent the message, the text of the message (a String), and a flag indicating whether or not the user has read the message. A mail system is a collection of users.

5. Write the data definitions for a mail system. Define a variable `mailsys` to be a mail system with no users.

6. Write a function `add-user` that consumes a username and produces void. The effect of the function is to add a new user with the given username to the mail system. The new user should have an empty mailbox. You may assume that the username is not already in the mail system.

7. Write a function `send-email` that consumes the name of the sender of an email, the name of the recipient of the email, and the text of an email message, and produces void. The effect of the function is to store a new unread message in the recipient's mailbox. Assume the named recipient is a user in the mail system.

8. Write a function `get-unread-messages` that consumes a username and produces a list of messages. The produced list contains the unread messages in the mailbox of the user with the given name. Assume the username is a valid user in the mail system. An effect of the function is that all unread messages in the named user's mailbox have been set to read.

9. Write a function `most-messages` that doesn't consume anything. The function produces the user in the mailsystem with the largest number of messages in his/her mailbox. If there are no users in the system, the function produces an appropriate error. If two or more users have the most messages, the function just needs to return one of them (it doesn't matter which one). You must use accumulator-style programming to solve this problem.

10. Show a sequence of test cases that you would use to test the email system.

What to Turn In

By now you should know what we expect from you on the homework, so the grade sheets are no longer available in advance.

Submit your `.rkt` file to InstructAssist. The name of the project is Homework 6. Follow the naming conventions for naming your file.
