Notes on Exam 2

The following notes pertain to your **second exam** in this course. No practice exam will be provided for the second exam. However, the structure of the second exam will be provided, and suggestions are made with respect to how you might want to study for the second exam.

1. **Your second exam will be administered on Wednesday, April 20th, 2022.** Details with respect to the administration of your second exam:
   (a) The exam will be given in the room where our class regularly meets, 170 Shineman, during the time when the class regularly meets, 11:30 am.
   (b) You will be given 60 minutes to write the exam.
   (c) No cognitive artifacts will be permitted for use on the exam – no books or notebooks or papers of any kind, beyond the examination itself; no phones or computers or other electronic resources.
   (d) Please come a few minutes early in order to settle into a proper frame of mind to take the exam.
2. **The structure of you second exam is suggested by the accompanying document.** As you can see, the exam will consist of six sets of questions, the first two pertaining programming with higher order functions in Racket, the last four pertaining to Prolog programming. With respect to the Racket programming, one set of questions will deal with higher order function interactions in the REPL, and the other will pertain to defining higher order functions. With respect to the Prolog programming, the first set of questions will be contextualized within the floating shapes KB, the second within the Pokemon KB, the third will pertain to list processing, and the fourth to state space problem solving and programming.
3. **Although you will not be provided with a practice exam, you should still engage in a well-focussed process of study for this exam**:
   (a) For the questions on higher order function interaction, you should prepare by presenting at least a couple dozen forms that feature higher order functions, especially `filter`, `map`, `foldr`, `foldl`, and `sort`, to the REPL. At least some of these should involve `lambda` expressions, since lambda expressions work so nicely with higher order functions. Where might you find forms involving higher order function for this purpose? The relevant lesson. The relevant programming assignment. Books. Google. Your mind.
   (b) For the questions on higher order function definition, you should start by reproducing and running all of the higher order function definitions provided in the relevant lesson, and all of those associated with the relevant programming assignment. First, if need be, reproduce them with the aid of whatever materials you may have at your disposal. Then work on doing so without materials, other than the models and processes in your mind. Of course, for each definition you write/rewrite you want to be sure to run the definition on the real machine to check your work. If time permits, search the world for some additional definitions to write and rewrite and test.
   (c) For the Prolog question pertaining to the floating shapes world, study the relevant part of the relevant lesson, redo the relevant part of the relevant programming assignment, and then (1) ask yourself basic questions about the KB (e.g., How many relations? How many facts?), (2) augment the KB with another shape and pose a few queries, (3) augment the KB with a few additional rules.
   (d) For the Prolog question pertaining to Pokemon, study the relevant part of the relevant lesson, redo the relevant part of the relevant programming assignment, and then (1) imagine/pose additional queries to the KB, and (2) augment the KB with a small number of additional rules.
   (e) For the Prolog question pertaining to list processing, work hard to get good at pattern matchine with head/tail notation. Review the relevant part of the relevant lesson. Redo the relevant part of the relevant

programming assignment. Type list equations involving head/tail notation, with some variables, into Prolog, until you are quite proficient at deconstructing lists and constructing lists by using the notation. Same for the recursive list processing! Review, redo, invent. Focus first on the recursive list processing notes provided in the relevant lesson and on the recursive list processors featured in the relevant programming assignment. Then, if time permits, explore more with the help of Google.

(f) For the Prolog question pertaining to state space problem solving, study the lesson on state space problem solving to the point where you know the definitions, the basic ideas associate with state space search, and are at least familiar with the problems presented in the lesson. Then focus your attention on the Towers of Hanoi state space programming assignment. I intend to ask you to do a little programming in the context of that assignment. For example, I might ask you to write one of the state space operator predicates, or I might ask you to write the predicate for checking for validity. Or, I might ask you to do a little something else.

4. **Please don't underestimate the importance of studying like you mean if for this exam. Proper preparation is key to success on an exam in which much of the material is foretold, at least to a considerable extent, and on which you are expected to demonstrate knowledge that you should have secured in your mind prior to arriving at the exam.**