Midterm Exam

Comp 210

Instructions

This exam is a three hour, take-home exam. The exam is closed book, closed notes. You may NOT use a computer on this exam. However, you may prepare and use one 8.5 x 11 crib sheet. Each student must prepare his or her own crib sheet. When done, please attach the crib sheet to your exam. You do not need to provide program schemas.

1. [15 points] Write a program that models an Ironman in the Beer-Bike competition as an object (without using global variables). An Ironman object is a function that accepts a symbol (representing an action) as its only argument. The valid actions are 'ride, 'chug, and 'intox-level.

   The 'chug action increments by one the number of beers currently ingested. (Initially, no beers have been ingested.)

   When invoked on the argument 'intox-level, the Ironman function returns the number of beers currently ingested.

   If an Ironman has currently ingested more than three beers, then the 'ride action causes the Ironman to puke, with the result that the number of beers ingested is reset to zero. Otherwise, the ride has no effect on the Ironman's internal state.

   "make-ironman": -3 (should have deducted more)

2. Recall that the function *list-max* applied to a list of non-negative numbers returns the value of the largest number in the list. (*list-max* returns 0 if the list is empty.)

   (a) [10 points] Implement *list-max* using a tail-recursive helper function that uses an accumulator. The helper function should be locally-defined.

   (b) [10 points] Convert this tail-recursive helper function into an equivalent *while* (also known as *while-fn*) loop (as described in class). You can assume that the definition of *while* loop has been previously provided. Again, all variables should be local to *list-max*.

3. (a) [5 points] Write a function *Insert* that takes two arguments — a number and a sorted list of numbers — and returns a sorted list that contains all the elements of the argument list as well as the argument number. In other words, *Insert* places a number in its proper location in a sorted list of numbers.

   (b) [10 points] Prove by induction that your function returns the appropriate sorted list. You may assume that the arguments to *Insert* are correct: a number and a sorted list of numbers. Recall that a list is sorted if each element (except the last) is less than or equal to the following element.