CS 115 – Dubai – Fall 2012

Assignment 06

Due: Thursday, November 15, 2012 at 11:59 p.m.

- Do **not** use reverse or member in any of your solutions.
- You may want to include defined constants to help reduce the writing for the examples and tests.
- For this and all subsequent assignments, you are expected to use the design recipe when writing functions from scratch. Be sure to follow all the steps of the design recipe, including the definition of constants and helper functions that include the design recipe, where appropriate.
- Do not copy the purpose directly from the assignment description. The purpose should be written in your own words and include references to the parameter names of your functions.
- The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.
- Do not send any code files by email to any course staff. It will not be accepted by course staff as an assignment submission. Course staff will not debug code emailed to them.
- Test data for all questions will always meet the stated assumptions for consumed values.
- Read each question carefully for restrictions.
- Read the course Web page for more information on assignment policies and how to organize and submit your work. Follow the instructions in the style guide. Specifically, your solutions should be placed in files a06qY.rkt, where Y is a value from 1 to 3.
- Download the interface file from the course Web page.

Language level: Beginning Student with List Abbreviations
Coverage: Module 7

1. A set in mathematics is defined as a collection of unique values. The union of two sets includes all numbers in each set, but where the values that are in both sets are only included once. The union of three sets of numbers would include all numbers in all three sets, but where the values appear in more than one set, they would appear only once in the union.

   Write a function called `union` that consumes three lists of numbers (`set-A`, `set-B`, and `set-C`) where the lists consumed are in sorted, ascending order. The function should produce a list that represents the union of these three sets, in ascending order. For example `(union (list 1 3 5) (list 1 2 5 6) (list 2 5 8 11))` produces `(list 1 2 3 5 6 8 11)`. The union of an empty list with a list A, would be the list A.

2. Ryder Cup is a four-day competition between golfers from the United States and golfers from Europe. Part of the competition includes individual match-play events. In match-play, the total score of the individual golfer does not matter – the results are determined by how many holes are won. A golfer wins a hole, if he/she has a lower score than the other golfer. In match-play, the player who wins the most number of holes, wins the match. If the players win the same number of holes, then the match is said to be "halved". The holes where the golfers shoot the same score do not factor into the final result.

   Write a function called `ryder-cup-result` that consumes two lists of natural numbers (`europe` and `usa`). The consumed lists represent the golf scores for several holes for a player from Europe and a player from USA. The function produces the string `"Europe"` if the `europe` scores result in a match-play win for that player, produces the string `"USA"` if the `usa` scores result in a match-play win for that player, and produces `"Halved"` if the scores from `europe` and `usa` result in a tie in match-play. You may assume that the lists are the same length.

For example `(ryder-cup-result (list 3 4 3) (list 4 3 3))` produces `"Halved"`, but `(ryder-cup-result (list 2 3 2 5) (list 3 2 3 5))` produces `"Europe"`.

3. Recall that the built-in function `substring` consumes a string, and two index positions (*start* and *end*), and produces a string from index position *start* to index position *end-1*. Write a function called `sublist` that will produce a similar result for a list. The function consumes a list (`whole-list`) containing any type of element and two natural numbers (`start` and `end`). The function produces a sublist beginning at position `start` and including all elements up to, but not including element at position `end`. If `start` and `end` are the same value, then the function will produce the empty list. The first element of the list is considered to be at position `0` in the list.

   You may assume that `start` is less than the length of the list, and that `end` is greater than or equal to `start` and less than or equal to the length of the list. For example, `(sublist (list 1 2 3 4 5) 2 4)` produces `(list 3 4)`.