CS 115 – Dubai – Fall 2012

Assignment 09

Due: Thursday, December 6, 2012 at 11:59 p.m.

- This assignment is testing concepts from Module 10.
- All helper functions must be defined as local functions to the main function. Remember that the Style Guide only requires a contract and a purpose for locally defined functions.
- You may also define local constants in any of your solutions.
- You must **not** use recursion in any of your solutions. All solutions must include at least one of the abstract list functions `map`, `filter`, or `foldr` used in a non-trivial way. It may be necessary to use more than one of these functions to complete the solutions. Any solutions that include recursion may receive 0 marks for correctness.

- Do **not** use reverse in any of your solutions.
- You may want to include defined constants to help reduce the writing for the examples and tests. These may be defined globally.
- For this assignment, 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 a09qY.rkt, where Y is a value from 1 to 3.
- Download the interface file from the course Web page.

Language level: Intermediate Student
Coverage: Module 10

Questions 1 and 2 on this assignment are repeated from previous assignments. However in this case, you must use abstract list functions, and not recursion to complete the solution. You may use the same contracts, purposes, examples, and tests from your previous solutions.

1. Using abstract list functions, write a function called `calc-sqrt` that consumes a list of numbers (`lon`) and produces a list of elements, where each element in the list produced is one of:
   - the positive square root of the number in `lon`, where the number in `lon` is a non-negative integer and its square root is also an integer
   - the symbol `'irrational`, where the number in `lon` is a positive integer, but its square root is a non-integer
   - the symbol `'imaginary` for all negative numbers in `lon`

   In the case where the element is a positive, non-integer, there is no matching entry in the list that is produced. The list produced contains the matching values in the same relative order as the original list.

For example,
```
(calc-sqrt (list 4 0 1.3 5 -1 -1.3))
```
produces
```
(list 2 0 'irrational 'imaginary 'imaginary)
```

2. Module 6 describes an association list (`al`) as a way of implementing a dictionary. Recall that association lists should have unique keys, but the values may be duplicated. Using abstract list functions, write the function `matching-values` that consumes an association list (`dictionary`) and a string (`s-value`) and produces list of keys from all the key/value pairs in `dictionary` that have a value that matches `s-value`. If there is no key/value pair with a value that is equal to `s-value`, then the function should produce an empty list. In the list that is produced, the keys should be in the same relative order as they were in `dictionary`. For example
```
(matching-values (list (list 1 "a") (list 3 "b") (list 4 "c")
                       (list 8 "a") (list 10 "a") (list 12 "b")) "a")
```
produces
```
(list 1 8 10).
```

3. The Caesar Cipher is an ancient cryptographic scheme where each letter of a message is shifted by a fixed amount. The letters of the original message are shifted to a later letter in the alphabet. If the shift amount causes a letter to be shifted beyond the letter `Z`, then the letters loop back to the beginning of the alphabet. For example, if the shift amount is `5`, then all occurrences of the letter `A` would be shifted to `F`, all occurrences of the letter `B` would be shifted to `G`, …, all occurrences of the letter `Z` would be shifted to `E`. You can read more information about the Caesar Cipher here: http://en.wikipedia.org/wiki/Caesar_cipher

   Also, to avoid cues about the original message, all non-alphabetic characters are removed from the message, and all letters in the coded message appear as uppercase letters.

   Using abstract list functions, write a function `caesar-cipher` that consumes a string (`message`) and a natural number (`shift`) that is less than `26`. The function will consume the message that will be encoded according to the shift amount given. However, the encoded message will only include uppercase letters that match the alphabetic letters that originally appeared in message.

   For example `(caesar-cipher "I love CS!" 5)` produces `"NQTAJHX"` and `(caesar-cipher "-5 ?" 3)` produces `""`.

   Notes:

   - The built-in function `char->integer` consumes a character and produces an integer in the range 0 to 255 that matches the ASCII code value of the character consumed. For example `(char->integer #\Z)` produces `90`.The built-in function `integer->char` produces a character

that is equal to the ASCII code consumed. For example `(integer->char 66)` produces `#\B`. These functions may be helpful.

- There are many functions involving strings and characters in the String documentation file that may be helpful, including functions that allow you to compare two characters to see if one character is less than or greater than another character.
- It may be helpful to define local constants to remember intermediate values as you try to compute the encoded message.