Homework 1 — assigned Tuesday 30 January — due Friday 9 February

General instructions

Prepare a single file homework1.hs containing all the requested function definitions.

1.1 Simple functions on numbers (10pts)

1. (10pts) Write a Haskell function `test :: Int -> Int -> Bool` that takes two integers and returns `True` if and only if the two integers are both odd.

1.2 List manipulation (30pts)

1. (10pts) Write a Haskell function `stutter :: [Char] -> [Char]` that takes a list of elements and returns a list where every element has been duplicated. For example, `stutter "Hello World"` evaluates to `"HHeellllloo  WWoorrlldd"`.
2. (10pts) Write a Haskell function `compress :: [Char] -> [Char]` that eliminates consecutive duplicate elements of a list. For example, `compress "HHeellllloo  WWoorrlldd"` evaluates to `"Hello World"`.
3. (10pts) Write a Haskell function `zipSum :: [Int] -> [Int] -> [Int]` that takes two equally sized lists of ints and returns a single list of ints in which each element at a given index is the sum of the corresponding values at that index from the input lists. For example, `zipSum [1,2,3] [4,5,6]` evaluates to `[5,7,9]`.

1.3 Using lists for sets: writing recursive functions over lists (40pts)

Let us use the Haskell type [Integer] to represent sets of integers. The representation invariants are that there are no duplicates in the list, and that the order of the list elements is increasing.

Do not use any of the built-in Haskell functions that manipulate lists as sets.

1. (10pts) Write a Haskell function `setUnion :: [Integer] -> [Integer] -> [Integer]` that takes two sets and returns their union.
2. (10pts) Write a Haskell function `setIntersection :: [Integer] -> [Integer] -> [Integer]` that takes two sets and returns their intersection.
3. (10pts) Write a Haskell function `setDifference :: [Integer] -> [Integer] -> [Integer]` that takes two sets and returns their set difference.
4. (10pts) Write a Haskell function `setEqual :: [Integer] -> [Integer] -> Bool` that takes two sets and returns `True` if and only if the two sets are equal.

1.4 More involved functions on numbers (20pts)

1. (20 pts) Quoting from Wikipedia, `https://en.wikipedia.org/wiki/Digital_root`:

   > The *digital root* of a non-negative integer is the (single digit) value obtained by an iterative process of summing digits, on each iteration using the result from the previous iteration to compute a digit sum. The process continues until a single-digit number is reached.

   Write a Haskell function `dr :: Integer -> Int` that takes an integer and computes its digital root. For example, `dr 65536` evaluates to `7`.

How to turn in

Use the UNM Learn facility as follows: Navigate to `https://learn.unm.edu/` and log in. Then click on CS-357L-000 (Spring 2018). Now click on Assignments in the left side navigation menu. After that, click on the appropriate homework assignment link. Now attach your `.hs` file(s) and click submit. You are allowed to submit as many times as you like but only the latest submission will be graded.