Homework 1 — assigned Thursday 2 September — due Friday 10 September [revised 7 September]

General instructions

Your programs must be prepared as Haskell scripts `hw1`*n*`.hs` (where *n* is 1, 2 or 3). You must provide adequate comments and you must test the code. Supply the files `hw1`*n*`.testlog.txt` to document your testing.

1.1 List manipulation (10pts)

Write a function `swap :: [a] -> [a]` that takes a list and returns the same list with each pair of elements swapped (skipping the middle element in case there is an odd number of elements) and the entirety reversed.

For example, `swap [1, 2, 3, 4, 5]` evaluates to `[4, 5, 3, 1, 2]`,
and `swap [1, 2, 3, 4, 5, 6, 7]` evaluates to `[6, 7, 3, 4, 5, 1, 2]`.

1.2 Numbers (40pts)

1. (15pts) Write a function `mp :: Int -> [Int]` such that `mp n` returns a list consisting of the first `n` Mersenne primes. Test this function for `n` up to 8.
2. (25pts) What do you need to do in order to compute a few more Mersenne primes, for instance up to 11? Put your answer in `hw12a.txt`, `hw12a.pdf`, `hw12a.hs`, and/or `hw12a.lhs`, as you see fit.

1.3 Using lists for sets: writing recursive functions over lists (50pts)

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.

5. (10pts) Write a Haskell function `powerSet: [Integer] -> [[Integer]]` that takes a set S and returns its powerset 2^S. (The powerset 2^S of a set S (sometimes written $P(S)$) is the set of all subsets of S.) Note that the result uses the type `[[Integer]]` to represent sets of sets of integers. Here the representation invariant is that there are no duplicates in the list; the order of the sublists is immaterial.

How to turn in

Submission instructions: use the turnin facility provided by the department; see course wiki for details.

Include the following statement with your submission, signed and dated:
I pledge my honor that in the preparation of this assignment I have complied with the University of New Mexico Board of Regents' Policy Manual.