Homework 5

CSC 121-2
Fall 2019

1. Using `map`, `foldl`, or `foldr`, write a one-line function `ordlist :: [Char] -> [Int]` that takes a list of characters and returns a list of the integer codes of those characters. For example, `ordlist ['A','b','C']` returns [65,98,67].

2. Using `map`, `foldl`, or `foldr`, write a one-line function `multpairs :: Num a => [(a,a)] -> [a]` that takes a list of pairs of numbers and returns a list of the products of each pair. For example, `multpairs [(1,2),(3,4)]` returns [2,12].

3. Using `map`, `foldl`, or `foldr`, write a one-line function `xor :: [Bool] -> Bool` that returns the exclusive OR of all the elements of the list. An empty list should give `False`.

4. Using `map`, `foldl`, or `foldr`, write a one-line function `duplist :: [a] -> [a]` that returns the same list with each element repeated twice in a row. For example, `duplist [1,2,3]` gives [1,1,2,2,3,3].

5. Represent a polynomial using a list of its coefficients, starting with the constant coefficient and going only as high as necessary. For example `[1,5,3]` represents the polynomial $3x^2 + 5x + 1$. Using `map`, `foldl`, or `foldr`, write one-line function `eval` that takes a polynomial represented this way and a value for x and returns the value of the polynomial at the given x. For example, `eval [1,5,3] 2` should give 23.

6. Let us consider an implementation of sets as lists, where each element of a set appears exactly once in a list and the elements appear in no particular order. Do not assume you can sort the lists. Do assume the input lists have no duplicate elements, and do guarantee the output lists have no duplicate elements. Using this implementation, and using `map`, `foldl`, or `foldr`, write one-line curried function to test whether an element is a `member` of a set.

7. Using `map`, `foldl`, or `foldr`, write one-line curried function to construct the `union` of two sets.

8. Using `map`, `foldl`, or `foldr`, write one-line curried function to construct the `intersection` of two sets.