Homework 6

CSC 121-2
Fall 2019

1. Using `map`, `foldl`, or `foldr`, write a one-line function `sqsum :: [Int] -> Int` that takes a list of integers and returns the sum of the squares of those integers. For example, `squarelist [1,2,3,4]` evaluates to 30. (Do not use the `sum` function.)

2. Using `map`, `foldl`, or `foldr`, write a one-line function `truecount :: [Bool] -> Int` that takes a list of boolean values and returns number of `True`s in the list.

3. Using `map`, `foldl`, or `foldr`, write a one-line function `maxpairs :: [(Int,Int)] -> [Int]` that takes a list of pairs of integers and returns the list of the maximum elements of each pair. For example, `maxpairs [(1,3),(4,2),(-3,-4)]` evaluates to [3,4,-3].

4. Using `map`, `foldl`, or `foldr`, write a one-line function `max :: [Int] -> Int` that returns the largest element of a list of integers. Your function need not behave well if the list is empty.

5. Using `map`, `foldl`, or `foldr`, write a one-line function `min :: [Int] -> Int` that returns the smallest element of a list of integers. Your function need not behave well if the list is empty.

6. Using `map`, `foldl`, or `foldr`, write a one-line function `convert :: [(a,b)]->([a],[b])` that converts a list of pairs into a pair of lists, preserving the order of the elements. For example, `convert [(1,2),(3,4),(5,6)]` evaluates to ([1,3,5],[2,4,6]).

7. Write a one-line function `map'` using only `foldl` or `foldr` that acts just like the library `map` function.