Functional Programming (Sheet #4)

Marius Gavrilescu

1. `foldl (-) x xs = (...((x - x0) - x1)... -xn)=x-x0-x1-...-xn=x-sum xs`

   `foldr (-) x xs` has alternating + and - signs, so it is not equal to the `x - sum xs`.

2. ```
   isort :: Ord a => [a] -> [a]
   isort [] = []
   isort (x:xs) = insert x (isort xs)

   insert :: Ord a => a -> [a] -> [a]
   insert x [] = [x]
   insert x (y:ys)
       | x < y = x:y:ys
       | otherwise = y : insert x ys
   ```

3. ```
   dedup :: Eq a => a -> [a] -> [a]
   dedup x [] = [x]
   dedup x (y:ys)
       | x == y    =   y:ys
       | otherwise = x:y:ys

   remdups :: Eq a => [a] -> [a]
   remdups = foldr dedup []
   ```

4. `scanl (/) 1 [1..n]` produces `[1, 1/1, 1/1/2, 1/1/2/3, ..., 1/1/2/.../n]`.

   `scanl1 (+) (scanl (/) 1 [1..n])` produces
   `1, 1+1/1, 1+1/1+1/1/2, ...1+1/1+1/1/2+...1/1/2/.../n`.

5. ```
   type Matrix a = [[a]]
   type IntMat = Matrix Integer

   scale :: Integer -> IntMat -> IntMat
   scale x matrix = map (\ row -> map (\ cell -> cell * x) row) matrix

   addMat :: IntMat -> IntMat -> IntMat
   addMat a b = zipWith addRow a b

   addRow :: [Integer] -> [Integer] -> [Integer]
   addRow = zipWith (+)

   transpose :: Matrix a -> Matrix a
   transpose ([]:_) = []
   transpose mat = (map head mat) : transpose (map tail mat)

   multMat :: IntMat -> IntMat -> IntMat
   multMat a b = map (\ i -> map (\ j -> dotp (a !! i) (bt !! j)) [0..(length(head a) - 1)]) [0..(lengt
       where bt = transpose b

   dotp :: [Integer] -> [Integer] -> Integer
   dotp xs ys = sum (zipWith (*) xs ys)
   ```
   ```

6.
```
showMat :: IntMat -> String
showMat rows = showMat' rows (map colWidth [0..(length (rows) - 1)])
    where colWidth id = maximum (map (\ row -> 1 + length(show(row !! id))) rows)

showMat' :: IntMat -> [Int] -> String
showMat' rows widths = concat (map (\ row -> showRow row widths) rows)

showRow :: [Integer] -> [Int] -> String
showRow [] _ = "\n"
showRow (x:xs) (y:ys) = replicate (y-length(show x)) ' ' ++ show x ++ showRow xs ys
```

7. (a) `taxicab = [a^3 + b^3 | a <- [0..], c <- [0..a-1], d <- [0..c-1], b <- [0..d-1], a^3 + b^3 == c^`

   (b) `taxicab !! 1` is 4104.

   (c) Seems so.