1. How would you add 1 to every element in a given matrix of integers? How would you sum the elements of a matrix? The function *zipWith* (+) add two rows, but what function would add two matrices? How would you define matrix multiplication?

   Solution:

   (a) $map\ (map\ (+1))$

   (b) $sum \circ map\ sum$

   (c) $zipWith\ (zipWith\ (+))$

   (d)

2. What are the dimensions of the matrix $[[\,],[\,]]$? Of the matrix $[\,]$?

   The function *cols* (here renamed as *transpose*) was defined by

   $$\begin{aligned} &transpose && :: [[a]] \to [[a]] \\ &transpose\ [xs] && = [[x] \mid x \leftarrow xs] \\ &transpose\ (xs : xss) && = zipWith\ (:)\ xs\ (transpose\ xss) \end{aligned}$$

   Fill in the dots that would enable you to replace the first clause by

   $$transpose\ [\,] = ...$$

   The above definition of *transpose* proceeds row by row. Here is part of a definition that proceeds column by column:

   $$transpose\ xss = map\ head\ xss : transpose\ (map\ tail\ xss)$$

   Complete the definition.

3. Which of the following equations are true (no justification is necesary):

   $$any\ p = (\neg) \circ all\ (\neg\ p)$$
   $$any\ null = null \circ cp$$

4. Given a function $sort :: (Ord\ a) \Rightarrow [a] \to [a]$ that sorts a list, construct a definition of

   $$nodups :: (Ord\ a) \Rightarrow [a] \to Bool$$

5. The function $nub :: (Eq\ a) \Rightarrow [a] \to [a]$ removes duplicates from a list (a version of this function is available in the library *Data.List*). Define *nub*. Assuming the order of the elements in the result is not important, define

   $$nub :: (Ord\ a) \Rightarrow [a] \to [a]$$

   so that the result is a more efficient function.

6. The functions *takeWhile* and *dropWhile* satisfy

   $$span\ p\ xs = (takeWhile\ p\ xs, dropWhile\ p\ xs)$$

   give direct recursive definitions of *takeWhile* and *dropWhile*.

   Assuming $whiteSpace :: Char \to Bool$ is a test for whether a character is white space (such as a space, a tab or a newline) or a visible character, construct a definiion of

   $$words :: String \to [Word]$$

   that breaks a string up into a list of words.

7. Define $minimum :: (Ord\ a) \Rightarrow [a] \rightarrow a$.