Declarative Programming

Workshop exercises set 3.

QUESTION 1
If you were working on a program that functioned as a web server, and thus its output was in the form of web pages, you could:

(a) have the program write out each part of the page as soon as it has decided what it should be;
(b) have the program generate the output in the form of a string, and then print the string;
(c) have the program generate the output in the form of a representation such as the HTML type of the previous questions, and then convert that to a string and then print the string.

Which of these approaches would you choose, and why?

QUESTION 2
Implement a function ftoc :: Double -> Double, which converts a temperature in Fahrenheit to Celsius. Recall that C = (5/9) * (F - 32).
What is the inferred type of the function if you comment out the type declaration? What does this tell you?

QUESTION 3
Implement a function quadRoots :: Double -> Double -> Double -> [Double], which computes the roots of the quadratic equation defined by
0 = a*x^2 + b*x + c, given a, b, and c. See
http://en.wikipedia.org/wiki/Quadratic_formula for the formula.
What is the inferred type of the function if you comment out the type declaration? What does this tell you?

QUESTION 4
Write a Haskell function to merge two sorted lists into a single sorted list

QUESTION 5
Write a Haskell version of the classic quicksort algorithm for lists.
(Note that while quicksort is a good algorithm for sorting arrays, it is not actually that good an algorithm for sorting lists; variations of merge sort generally perform better. However, that fact has no bearing on this exercise.)

QUESTION 6
Given the following type definition for binary search trees from lectures,

```
>data Tree k v = Leaf | Node k v (Tree k v) (Tree k v)
>        deriving (Eq, Show)
```

define a function

```
>same_shape :: Tree a b -> Tree c d -> Bool
```

which returns True if the two trees have the same shape: same arrangement of nodes and leaves, but possibly different keys and values in the nodes.

QUESTION 7
Consider the following type definitions, which allow us to represent expressions containing integers, variables "a" and "b", and operators for addition, subtraction, multiplication and division.

```
>data Expression
>       = Var Variable
>       | Num Integer
>       | Plus Expression Expression
>       | Minus Expression Expression
>       | Times Expression Expression
>       | Div Expression Expression
```

```
>data Variable = A | B
```

For example, we can define exp1 to be a representation of 2*a + b as follows:

```
>exp1 = Plus (Times (Num 2) (Var A)) (Var B)
```

Write a function eval :: Integer -> Integer -> Expression -> Integer which takes the values of a and b and an expression, and returns the value of the expression. For example eval 3 4 exp1 = 10.