Concepts in Programming Languages

Supervision 2

3. Scripting and modularity

1. What is a scripting language? What kinds of applications *should* scripting languages be used for? In contrast, what kinds of applications *are* they used for today?
2. Compare and contrast *duck typing* and *dynamic typing*. Why do they tend to be features of scripting languages?
3. Compare and contrast module systems with principles of object-oriented programming such as abstraction and inheritance. How do ML structures and signatures differ from OOP classes and interfaces? Compare these with Haskell's type classes, if you are familiar with them.
4. a) Write a signature for a Queue abstract data type in Standard ML.
   b) Write two structures implementing this signature: one using a single list, and one using a pair of lists (with amortised constant time for its operations, as covered in Foundations of Computer Science). You should use the same kind of signature constraint for both of them.
   c) Did you use an opaque or transparent signature constraint? What difference does it make?

4. Further concepts

1. You manage two junior programmers and overhear the following conversation:

   Alice: "I don't know why anyone needs a language other than Java, it provides clean thread-based parallel programming."

   Bob: "Maybe, but I write my parallel programs in a functional programming language because they are embarrassingly parallel."

   Discuss the correctness of these statements and the extent to which they cover the range of languages for parallel programming. *Note*: this was an exam question in 2014, *after* the release of Java 8 – make sure to consider this in your answer to the first part of the question.
2. Describe the *expression problem* and how it relates to data types in functional and object-oriented languages, with the help of code examples. How can both of the "competing" approaches be implemented in Scala? Can you think of a time when you encountered the expression problem in your own projects?
3. Reason for and/or against the following statement:

   Functional programming is the future.

Some questions by Andy Rice (acr31) and Andrej Ivašković (ai294).

4. a) What is a *monad*? What are its operations?
   b) Distinguish between a side-effecting function, a pure function, and a "computation" value in a monad.

5. Assume the existence of an IO monad in a functional language.

   a) Give the types of expressions which:
      (i) read a line from `stdin`;
      (ii) read a line from a file specified by parameter `f`;
      (iii) write a line to `stdout`;
      (iv) write a line to a file specified by parameter `f`.
   b) Given values `c : unit IO` and `n : int`, give a program which performs `c`
      (i) twice;
      (ii) `n` times.

6. Suppose you want to define a datatype for lists containing *alternating* integers and Booleans, e.g. `[5, true, 2, false, 9, true]`. How can you statically enforce this alternation property using GADTs?

7. Briefly explain the intuition behind using CPS to represent a function of type `a -> b` as `(b -> unit) -> (a -> unit)`. How and why does contravariance come into play?