1 Concepts in Programming Languages (am21)

(a) Consider a program in a simple language \( L \) which manipulates only two forms of values: 64-bit two’s-complement integers and 64-bit IEEE floating-point values. \( L \) includes variables, integer and floating-point operators and constants, and conditional expressions, but no functions. It is considered a (hard) error if a bit-pattern representing a floating-point value is operated on as if it were an integer and vice versa. There are no implicit coercions, so \( 1 + 2.3 \) would be an error. The job of both static and dynamic type checkers is to stop such errors from happening.

(i) Explain the difference between static type checking and dynamic type checking, pointing out any compile-time or run-time costs and any differences in how variables are declared. [3 marks]

(ii) We can say that a static type system is **sound** if whenever we have two programs \( S \) and \( D \), differing only in whether they use static or dynamic type checking, then \( S \) passing type checking implies \( D \) executes successfully. In a sound type system, does \( D \) executing successfully imply that \( S \) passes type checking? Justify your answer for \( L \). [2 marks]

(b) Give three programs exemplifying failure of type soundness in existing languages. Two should involve distinct past-or-current programming languages and one should reflect the absence of checks performed by most linkers. [4 marks]

(c) To what extent does Java use static and/or dynamic typing? [2 marks]

(d) (i) A monad \( M \) can be seen as a type constructor for an abstract data type. Give the two operations which every monad must possess, along with their types. [3 marks]

(ii) Now consider the monad \( E \) and the functions \( f \) and \( \text{safediv} \), given in SML syntax by:

\[
\begin{aligned}
\text{datatype a E = return a | fail} \\
\text{fun (fail >>= f) = fail} \\
\quad | ((return v) >>= f) = f v \\
\text{fun f(w,x,y,z) = (w div x) div (y div z)} \\
\text{fun safediv(x,y) = if y<>0 then return(x div y) else fail}
\end{aligned}
\]

Re-code \( f \) as a function \( g \) which uses operations from monad \( E \) along with function \( \text{safediv} \) so that the only non-zero tests around division are within \( \text{safediv} \). Remark on any difference between the types of \( f \) and \( g \). [6 marks]