Solution notes

Comparative Programming Languages 2005 (MR)
Paper 6 Question 7, Paper 13 Question 7

Data types are covered in many parts of the course with many examples of type system from untyped through to very strictly typed languages. Students should have a good understanding of the relative merits of all of these. Students should demonstrate that they have a good understanding of the advantages and disadvantages of the schemes mentioned in the question, but their actual preference will not affect the mark.

a) Strong compile time typing allows certain kinds of programming errors to be found early hence programs written in such languages may be more reliable and more easily debugged. A drawback is that type systems are often complicated and no fully understood by most programmers, possibly making programs longer and harder to write. Types often help the reader (and maintainer) of a program to understand it.

b) As has been seen in PL/1, having too many numerical types leads to problems. It becomes neccesary to have automatic type conversion to make programming convenient, but the convention rules become much more complicated (and arbitrary) the more numerical types there are. This often leads to anomilies such as the PL/1 expression 9+8/3 yielding either overflow or 1.6666... both of which are arguably wrong, although they obey the seemingly sensible type convertion rules to the letter. What advantage would there be to having a data type for distance. Would it report an error if the second argument of a function were a distance rather than a time? Could the second argument be written as 13.7 or now+1? If x were of type distance, what would be the type of x*x, x*x*x, etc. Would you be able to write x+1? Could you have a distance in inches, centimetres, miles, light years etc. Would the typing system do the right thing when distances in different units were combined. Would the every day programmers thoroughly know and understand the rules. Sounds like considerable room for complexity, ambiguity, misunderstanding, and anomoly.

c) When compiling a function call, the compiler must know the mode of calling of the arguments so this must be specified in some way. Being part of the functions data type is certainly one way. A less efficient alternative is to carry this information as part of the runtime representation of a function and interpret it every time the function is called. This is still more complicated if the language allow variadic or polymorphic functions. Because of these complications and the availability of pointers and functions as first class types most modern languages always call function arguments by value. In Algol W, the mode of calling arguments is not required because they are, in effect, all called by name, since the calling modes are defined in terms of rewrite rule on the bodies of procedures. Needless to say, a good compiler will often compile good code for modes such as call-by-value when it can tell this better code is safe.