COMPUTER SCIENCE TRIPOS Part IA – 2009 – Paper 1

1 Foundations of Computer Science (LCP)

This question has been translated from Standard ML to OCaml

(*a*) The polymorphic curried function `delFirst` takes two arguments, a predicate (Boolean-valued function) `p` and a list `xs`. It returns a list identical to `xs` except that the first element satisfying `p` is omitted; if no such element exists, then it raises an exception. Code this function in OCaml. [4 marks]

(*b*) Use the function `delFirst` to express the polymorphic function `delFirstElt`, where `delFirstElt x xs` returns a list identical to `xs` except that it omits the first occurrence of `x`. [2 marks]

(*c*) Carefully explain the polymorphic types of these two functions, paying particular attention to currying and equality. [4 marks]

(*d*) A list `ys` is a *permutation* of another list `xs` if `ys` is obtained by rearranging the elements of `xs`. For example, `[2; 1; 2; 1]` is a permutation of `[2; 2; 1; 1]`. Code an OCaml function to determine whether one list is a permutation of another. [4 marks]

(*e*) A list `ys` is a *generalised permutation* of `xs` if `ys` is obtained by rearranging the elements of `xs`, where one element of `xs` is specially treated: it may appear any number of times (including zero) in `ys`. For example, `[1; 2; 1]` is a generalised permutation of `[1; 2]` but `[1; 2; 2; 1]` is not because two elements (1 and 2) appear the wrong number of times in it. Code an OCaml function to determine whether one list is a generalised permutation of another. [6 marks]

All OCaml code must be explained clearly.