COMPUTER SCIENCE TRIPOS Part IA – 2025 – Paper 1

1 Foundations of Computer Science (avsm2)

(*a*) Given the following incorrect code that implements a merge sort, identify five errors and suggest corrections with a brief explanation to make it work.

```
let length = function
  | [] -> 0
  | _ :: t -> 1 + length t

let rec merge l1 l2 =
  match l1, l2 with
  | [], l -> l
  | h1 :: t1, h2 :: t2 ->
      if h1 <= h2 then h1 :: merge t1 l2
      else h2 :: merge l1 t2

let rec split l l1 l2 n =
  match l, n with
  | [], _ -> (l1, l2)
  | h :: t, 0 -> split t l1 (h :: l2) n
  | h :: t, _ -> split t (l1 :: h) l2 n

let rec mergesort ls =
  match ls with
  | _ ->
      let n = length ls / 2 in
      let l, r = split ls [] [] n in
      merge (mergesort l) (mergesort r)
  | [] | [_] -> ls
```

[10 marks]

(*b*) Write a function `val check_sorted : 'a list -> bool` that returns `true` if the input list is already sorted, and define its time and space complexity. *(Hint: you can use the polymorphic* `<=` *operator here).* [4 marks]

(*c*) You find a library that implements `quicksort`, `bubblesort` and `insertsort` functions that all have the same type as `mergesort`. Write a function:

```
val checksort: ('a -> 'b list) list -> 'a -> unit
```

`checksort` should apply the input sorting functions and check that each result is sorted. Define a mechanism to signal if an unexpected mismatch is discovered. Give an example of using `checksort` to verify that `quicksort`, `bubblesort` and `insertsort` work for some sample integer list. [6 marks]