Foundations of Computer Science – Supervision 1

Question 1: (a) Why do we need types in programming languages?

- (b) What is the type of the following values? Briefly justify:
 - (i) let a x y = x
 - (ii) let b x y z = if x then y else z
 - (iii) let c x y = x y
 - (iv) let rec d = x = d [x] true
- (c) Define values with the following types (without using type annotations):
 - (i) 'a -> 'a
 - (ii) int -> int list
 - (iii) ('a -> 'b) -> ('b -> 'c) -> 'a -> 'c
 - (iv) ('a -> 'b) -> 'a list -> 'b list
- Question 2: (a) For each function below, state its behaviour and its time and space complexities. Assume no compiler optimisation (tail-recursion doesn't count as compiler optimisation). Justify your answers.

 - (ii) let rec g n = if n > 1 then g (n 1) + g (n 2) else n
- Question 3: (a) Write a function that takes a positive integer n and returns a list of its factors. For example: factors 12 = [1;2;3;4;6;12]. The order of the factors in the list can be arbitrary, and you can assume that the argument is always positive.
- (b) What is the time complexity of your algorithm? Can you think of a way to improve it?
- Question 4: Link to past paper question
- **Question 5:** (a) Implement the following utility functions on lists. Make sure that all recursive calls are tail-recursive:
 - (i) reverse:
 reverse [x1;...;xn] is [xn;...;x1]

- (ii) map: map f [x1;...;xn] is [f(x1);...;f(xn)]
- (iii) filter: filter p l is the list containing the elements of l that satisfy the predicate p, in the order that they appear in l.
- (iv) left fold:

```
fold_left f a [b1;b2;...;bn]
= f (... (f (f a b1) b2) ...) bn
```