Computer Science 716 **Theory Test**
Functional Programming: Haskell **20/10/2016**

A Theory

(1) Explain why `foldl (+) 0 (replicate 100000000 1)` (or slightly larger numbers than `100000000`) leads to a stack overflow (in ghci). How can one avoid this stack overflow by (a) using another library function, or (b) implementing your own version of `foldl`.

(2) Explain why it is a poor idea to use a Haskell list to represent the Queue abstract data type, where the front of the queue is the same as the front of the list.

B Programming

(1) Recall that the Fibonacci numbers are `1, 1, 2, 3, 5, 8` et cetera, where each Fibonacci is the sum of the preceding two. Write a Haskell function `fib` that takes an integer parameter `n` and computes the `n`th Fibonacci number in time `O(n)`. For example, `fib 5` should return `8`.

(2) Without using any identifiers to represent parameters, define a function `squareFn` that takes a function f of type `Double -> Double` and returns a new function g where $g(x) = (f(x))^2$.

(3) Write a function `mapEach :: (a -> b) -> [[a]] -> [[b]]` such that `mapEach f xss` returns a list of lists, containing the result of applying `f` to each element of each sublist of `xss`. For example,

```
mapEach (+ 3) [] = []
mapEach (+ 4) [[]] = [[]]
mapEach (+ 2) [[5, 4, 1], [7, 6], [8, 12]]
= [[7, 6, 3], [9, 8], [10,14]]
mapEach (> 3) [[1, 7], [0, 4], [9, 10], []]
= [[False,True],[False,True],[True,True],[]]
mapEach (\y -> y*y) [[0, 4], [9, 10, 2, 7, 5], [1 .. 3]]
= [[0,16], [81,100,4,49,25], [1, 4, 9]]
```