Exercise Sheet One: Getting Started with Haskell

The aim of this tutorial is to introduce you to the Haskell language. By the end of the class you should have a basic understanding of GHCi the Haskell interactive interpreter and you should have written at least one simple Haskell script.

Attempt the exercsise in your own time before Tuesday's lab and use that session for obtaining feedback and for asking questions related to the exercises.

If you have not yet installed Haskell on your own machine then you could visit Haskell Platform and do so. Alternatively you may use installation on the lab machines.

Exercise One

The function

```
double x = x + x
```

takes a number and returns twice its argument. To expand an expression `double E`, rewrite it with the right hand side of the function definition but with `x` replaced by the expression `E`. For example `double 10` could be expanded by replacing `x` in the definition above with `10` to get `10 + 10`. Write down an evaluation of the expression `double (double 2)` by repeatedly expanding the definition of `double` and `+`. You may assume that `+` behaves as expected. Is there more than one way of doing this? Is one way more efficient than the other?

Exercise Two

The function

```
sum [] = 0
sum (x:xs) = x + sum xs
```

takes a list of numbers and returns the total of adding all numbers in the list. This says, in the case of an empty list `[]` the value returned should be 0. For lists that comprise some element `x` followed by another (possibly empty) list `xs` the return value should be `x` added to the recursively evaluated sum on `xs`. By expanding the above definition show that `sum [x] = x` for any `x`. That is, summing the elements of the singleton list containing just `x` will return that value.

Exercise Three

Define a function `product` that produces the product of a list of numbers. Test your definition on a few examples.

Exercise Four

Recall the definition of Quicksort from the lecture:

```
quicksort [] = []
quicksort (x:xs) = quicksort ls ++ [x] ++ quicksort rs
                   where
                     ls = [ a | a <- xs , a <= x ]
                     rs = [ a | a <- xs , a > x ]
```

Modify this definition to create a function `quicktros` that produces a sorted version of the argument list in descending order.

Exercise Five

Suppose the definition of `quicksort` above used `<` instead of `<=` when defining `ls`. What would the effect of this be? See if you can work this out and then modify the code and test it on a few examples.

Exercise Six

Evaluate each of the following expressions in GHCi and then repeat each one by adding explicit parentheses in place of the implicit parentheses. Make sure the expressions evaluate to the same value in each case.

- `2^3*4`
- `2*3+4*5`
- `2+3*4^5`
- `2^2+2^2`

Exercise Seven

The following script contains at least three syntactic errors. Copy the code in to a script, correct all errors and check that your solution loads correctly using GHCi.

```
N = a 'div' length xs
    where
       a = 10
      xs = [1,2,3,4,5]
```

Exercise Eight

The libary function `last` selects the last element of a non-empty list; for example `last [1,2,3,4,5] = 5`. Show how the function `last` could be defined in terms of the other libary functions given in the lecture in at least two distinct ways.