**Department of Applied Mathematics and Computational Sciences**

**PSG College of Technology**

**Programme: MSc SS VIII semester**

**Course: Functional Programming Lab**

**PROBLEM SHEET 1**

**(exercises based on Programming in Haskell, Graham Hutton)**

**INTRODUCTION**

1. Give another possible calculation for the result of double (double 2).

double :: Num a => a -> a

double x = x + x

1. Show that sum[x] = x for any number x.

sum [x] = x

1. Define a function product that produces the product of a list of numbers, and show using your definition that product [2, 3, 4] = 24.

product :: Num a => [a] -> a

product [] = 1

product (x:xs) = x \* product xs

product [2, 3, 4]

= 2 \* product [3, 4]

= 2 \* (3 \* product [4])

= 2 \* (3 \* (4 \* product []))

= 2 \* (3 \* (4 \* 1))

= 2 \* (3 \* 4)

= 2 \* 12

= 24

1. How should the definition of the function *qsort* be modified so that it produces a *reverse* sorted version of a list?

qsort :: Ord a => [a] -> [a]

qsort [] = []

qsort (x:xs) = qsort [y | y <- xs, y <= x] ++ [x] ++ qsort [y | y <- xs, y > x]

qsort :: Ord a => [a] -> [a]

qsort [] = []

qsort (x:xs) = qsort [y | y <- xs, y >= x] ++ [x] ++ qsort [y | y <- xs, y < x]

1. What would be the effect of replacing ≤ by *<* in the definition of *qsort*? Hint: consider the example *qsort* [2*,* 2*,* 3*,* 1*,* 1].

Original qsort:

qsort [2, 2, 3, 1, 1]

= qsort [1, 1] ++ [2, 2] ++ qsort [3]

= [1, 1, 2, 2, 3]

Modified:

qsort [2, 2, 3, 1, 1]

= qsort [1, 1] ++ [2, 2] ++ qsort [3]

= [1, 1, 2, 2, 3]

**FIRST STEPS**

1. Parenthesise the following arithmetic expressions:

2 ↑ 3 x 4 : 2 ↑ (3 x 4)

2 x 3 + 4 x 5 : (2 \* 3) + (4 \* 5)

2 + 3 x 4 ↑ 5: 2 + (3 \* (4 ^ 5))

1. Work through the examples from this chapter using Hugs.
2. The script below contains three syntactic errors. Correct these errors and then check that your script works properly using Hugs.

N = a ’div’ length xs

where

a = 10

xs = [1, 2, 3, 4, 5]

N = a `div` length xs

where

a = 10

xs = [1, 2, 3, 4, 5]

1. Show how the library function last that selects the last element of a nonempty list could be defined in terms of the library functions introduced in this chapter. Can you think of another possible definition?

last :: [a] -> a

last [x] = x

last (\_:xs) = last xs

last :: [a] -> a

last xs = head (reverse xs)

1. Show how the library function init that removes the last element from a non-empty list could similarly be defined in two different ways.

init :: [a] -> [a]

init [x] = []

init (x:xs) = x : init xs

init :: [a] -> [a]

init xs = reverse (tail (reverse xs))