Synopsis Lab 8

8.1 Compiling Haskell programs

Copy the code below into `factori.lhs`. To compile and run it, you can choose between two methods:

1. type at the Linux shell prompt:

   ```
   runhaskell factori.lhs
   ```

2. use `ghc` to compile the code, by typing at the Linux shell prompt:

   ```
   ghc -O3 factori.lhs -o factori.exe
   ```

   and then run it with:

   ```
   ./factori.exe
   ```

```
\begin{code}
--declare module Main. This module contains the program entry point
module Main where

  factori n = fact_acc n 1

  fact_acc 0 a = a
  fact_acc n a = fact_acc (n-1) $! (n*a)

  a = 1
  b = 9
  c = 11
  d = 3

  n1 = (factori (a+b)) `div` (factori a)
  n2 = (factori (a+c)) `div` (factori c)
  n3 = (factori (b+d)) `div` (factori b)
  n4 = (factori (c+d)) `div` (factori d)

  numer = n1 * n2 * n3 * n4
  denom = factori (a+b+c+d)

  p = (fromIntegral numer) / (fromIntegral denom)
```

```
  -- program entry point
  main = do
          print p
\end{code}
```

8.2 Tasks

1. Skim lectures #4, #5 and #7. Test the following functions: `union`, `zip`, `unzip`, `zipWith`, `map`, `foldr`, function composition (`.`) (note: `map, filter, zip, unzip, foldr` are already defined in Haskell). Test the code concerning exceptions.

2. (Haskell, ML) Write a function called `commonFactors` with arguments `n1` and `n2` which returns the list of all common factors of `n1` and `n2`. E.g.:

   ```
   Main> commonFactors 12 18
   [1,2,3,6]
   ```

3. (ML) Write a function called `equation` having 2 parameters `a` and `b`, which solves the equation `ax+b=0` and raises an exception if `a=b=0` and another one if `a=0` and `b<>0`.

4. (Haskell, ML) Write a function called `innerProduct` with arguments `v1` and `v2` which computes the inner product of 2 vectors `v1` and `v2` given as lists of numbers:

   $$innerProduct\ x\ y = \sum_{i=1}^{n}(x_i * y_i)$$

   E.g.:

   ```
   Main> innerProduct [1,2,3] [2,4,3]
   19
   ```

   How many essentially different Haskell solutions can you provide?