CSA1080: Declarative Programming  

XXX June 2004

Answer two questions from section A, and two questions from section B. Each question carries 25 marks. Students are allowed to use course notes, books and calculators.
Section B: Functional Programming

4. (a) The following data type describes steps of a given length in one of the four compass directions:

```haskell
data Step = North Int
          | South Int
          | East Int
          | West Int

type Path = [Step]
```

Show your understanding of the above datatype by giving the type and defining the function `followPath`, which given `Path`, returns the x and y coordinates reached after following the given path. For example, `followPath [North 6, West 3, South 4]` would return `(2,-3)`. A step is said to be in normal form, if its size is positive. A step not in normal form can be made in normal form by switching the direction, and negating the size. For example, `North -4` can be normalised to `South 4`. Define a function `normalise` which normalises a step. Hence define a function which normalises a path.

What do the following three functions do? What are their types?

```haskell
dragonfly f (North n) = North (f n)
dragonfly f (South n) = South (f n)
dragonfly f (East n) = East (f n)
dragonfly f (West n) = West (f n)

butterfly f p = map (dragonfly f) p

briefly n = butterfly (n*)
```

(b) Give the type of the following two functions:

```haskell
whatsTheType1 y xs = [ y ++ map toLower x | x <- xs ]
```

Page 3 of 5
whatsTheType2 [] _ _ = []
whatsTheType2 _ _ "" = []
whatsTheType2 xs f ys =
   f ys ++ xs ++ whatsTheType2 (tail xs) f (tail ys)

5. (a) Consider the following definition:

   data Describe a = Description String a
                   | NoDescription a

   All objects of type Describe a can be transformed to a string by taking their first parameter. Therefore, Description "Pippo" x would be transformed to "Pippo". In the case of NoDescription, the result would be "Unknown". Give the necessary code to make Describe a an instance of the Show class.

   (b) Define a function commonPrefix, which given two lists, returns the common prefix of the lists. For example (commonPrefix [1,2,3] [1,3,7,2,3] would return [1].

      commonPrefix :: Eq a => [a] -> [a] -> [a]

      Hence or otherwise, define a function commonSuffix which returns the common suffix of two lists. For example (commonPrefix [1,2,3] [1,3,7,2,3] would return [2,3].

   (c) Assume you have a function which returns all proper divisors of a number:

      properDivisors :: Int -> [Int]

      Thus, for example, (properDivisors 6) would return [1,2,3]. A perfect number is a number whose sum of proper divisors is equal to the number itself. 6 is thus a perfect number, since it is equal to 1+2+3. Using properDivisors, write a list comprehension which produces all perfect numbers.
6. (a) Define a typeclass `BoolClass`, which includes types `a` supporting `negation` (a function from `a` to `a`) and `disjunction` (a function which takes two objects of type `a` and returns an object of type `a`).

Make `Bool` an instance of `BoolClass`.

Conjunction is defined as follows:

\[
\text{conjunction } v1 \ v2 = \\
\text{negation (disjunction (negation } v1 \ (\text{negation } v2))}
\]

Give the type of `conjunction`.

(b) Define a function `mylookup`, which given a list of pairs of type `(a,b)`, and an item of type `a`, returns all the objects of type `b` which are paired up with the input in the given list. For example, `mylookup [(1,'a'), (2,'c'), (1,'d')] 1)` would return `['a','d']`.

`mylookup :: Eq a => [(a,b)] -> a -> [b]`

Use `mylookup` to define a function which takes a list of objects of type `a` and returns all the matching ones. For example, `mylookups [(1,'a'), (2,'c'), (1,'d'), (3,'x')] [1,3]` would return `['a','d','x']` (in any order).

`mylookups :: Eq a => [(a,b)] -> [a] -> [b]`

(c) Using a list comprehension, write a function which calculates the divisors of a given number (divisors of a number `n`, are numbers `m`, for which `n ÷ m` leaves a remainder of zero).

`divisors :: Int -> [Int]`