Lund University
Department of Computer Science

EDAN40: Functional Programming
28th May 2018, 14–19

Exam

1. **Point-free notation**

   Rewrite the following two definitions into a point-free form (i.e., `f = ...`, `g = ...`), using neither lambda-expressions nor list comprehensions nor enumeration nor `where` clause nor `let` clause:

   ```
   f x y = (3 - y) / x
   g x y = [x z | z <- [1,3..y]]
   ```

2. **Type derivation**

   Give the type of the following expressions:

   (a) `(.)(:)`

   (b) `(:)(.)`

   (c) `((.):)`

   (d) `(:(.))`

   (e) (Haskell swearing) `([]>>=)(\_->[(>=)])`

3. **Proving program properties**

   The `Functor` class is defined as follows:

   ```
   class Functor f where
     fmap :: (a -> b) -> f a -> f b
   ```

   It is mandatory that all instances of `Functor` should obey:

   ```
   fmap id      = id
   fmap (p . q) = (fmap p) . (fmap q)
   ```

   Assume the following definition of lists as a functor instance:

   ```
   instance Functor [] where
     fmap g []     = []
     fmap g (x:xs) = g x : (fmap g xs)
   ```

   Is this a correct definition of a functor instance? Why or why not? **Prove your claim**.

4. **Programming**

   Give an example of a function with type

   ```
   ([a] , a -> b) -> [b]
   ```

5. **Type classes**

   Complete the following two instance declarations:

   ```
   instance (Ord a, Ord b) => Ord (a,b) where ...

   instance Ord b => Ord [b] where ...
   ```

   where pairs and lists should be ordered lexicographically, like the words in dictionary.

6. **Monadic computations**

   Given the following function:

   ```
   f x y = do
     a <- x
     b <- y
     return (a*b)
   ```

   (a) What is the type of `f`? (0.1)

   (b) What is the value of `f [1,2,3] [2,4,8]` ? (0.2)

   (c) What is the value of `f (Just 5) Nothing` ? (0.1)

   (d) What is the type of expression `return 5`? (0.1)

   (e) What is the value of expression `do [1,2,3]; []; "abc"`? (0.25)

   (f) What is the value of expression `do [1,2,3]; []; return "abc"`? (0.25)

Good Luck!