FUNCTIONAL PROGRAMMING MT2018

Sheet 2

3.1 Suppose that the class of ordered types is declared by something like

```
class Eq a => Ord a where
  (<), (<=), (>), (>=) :: a -> a -> Bool
  x <  y = not (x >= y)
  x >  y = not (x <= y)
  x >= y = x == y || x > y
```

(It includes a couple of other things, and these are not quite the default definitions.) Lists are lexicographically ordered, like the words of a dictionary. Write an instance declaration for `Ord [a]`.

3.2 Suppose $h\ x\ y = f\ (g\ x\ y)$. Which of the following are true, which are false, and (in each case) why?

1. $h = f \cdot g$
2. $h\ x = f \cdot g\ x$
3. $h\ x\ y = (f \cdot g)\ x\ y$

3.3 Give most general types for the following, where possible

```
> subst f g x = (f x) (g x)
> fix f = f (fix f)
> twice f = f . f
> selfie f = f f
```

You should try to work out what the most general type is by hand, but you can check that you are right by using an interpreter; and if you are wrong, check that you understand why.

3.4 Which of these equations are badly typed? For the others, what can you say about the type of xs, and whether and when the equation holds?

$a)\ [\,] : xs = xs$	$e)\ [\,] : xs = [[\,], xs]$	$i)\ xs : [\,] = xs$
$b)\ xs : [\,] = [xs]$	$f)\ xs : xs = [xs, xs]$	$j)\ xs : [xs] = [xs, xs]$
$c)\ [[\,]] +\!\!+ xs = xs$	$g)\ [[\,]] +\!\!+ xs = [xs]$	$k)\ [[\,]] +\!\!+ xs = [[\,], xs]$
$d)\ [[\,]] +\!\!+ [xs] = [[\,], xs]$	$h)\ [xs] +\!\!+ [\,] = [xs]$	$l)\ [xs] +\!\!+ [xs] = [xs, xs]$

4.1 Show that if f and g are strict, so is the composition $f \cdot g$.
Is the converse true: that if $f \cdot g$ is strict, so must f and g be?

4.2 If we count $\bot :: Bool$ as well as the proper values, there are three values of type $Bool$. So how many functions are there of type $Bool \to Bool$? How many of these are computable? Are all the computable ones definable in Haskell?

4.3 By evaluating expressions like `False && undefined` and other expressions using combinations of *True*, *False*, and *undefined* find exactly which function `(&&)` is implemented in the standard prelude. Give a definition which would produce that behaviour.

There is exactly one computable function, (&&&) say, which satisfies

$$\begin{aligned} False \mathbin{\&\&\&} y &= False \\ x \mathbin{\&\&\&} False &= False \\ True \mathbin{\&\&\&} True &= True \end{aligned}$$

for all x and y, including $\bot$. Given that it is computable, explain what this function does in other cases, and so why there is only one such function. Is it definable in Haskell?

4.4 Time (as Richard Bird might say) for a song:

```
One man went to mow
Went to mow a meadow
One man and his dog
Went to mow a meadow

Two men went to mow
Went to mow a meadow
Two men, one man and his dog
Went to mow a meadow

Three men went to mow
Went to mow a meadow
Three men, two men, one man and his dog
Went to mow a meadow
```

Write a Haskell function $song :: Int \to String$ so that $song\ n$ is the song when there are n men (and a dog). Assume $n \leqslant 10$. To print the song, type for example: `putStr (song 5)`. You may want to start from

```
> song 0  = ""
> song n  = song (n-1) ++ "\n" ++ verse n
> verse n = line1 n ++ line ++ line3 n ++ line
```

Geraint Jones, 2018