School of Computer Science, University of Nottingham

G51FUN Functional Programming, Spring 2015

Graham Hutton

Coursework I, Part 2/5

Monday 2nd February 2015

Deadline: Monday 16th February 2015, 11am

This exercise sheet covers material from Lecture 3 (Types and Classes), and is worth 20% of the first coursework. Half of the marks are awarded for your solution; the other half are awarded for your answers to the tutor's questions. You should attempt to complete the exercises in your own time, but if you get stuck you can ask for help during the lab sessions.

Assessment will be carried out by oral examination during the lab sessions (nothing needs to be handed in). When you have completed the exercises ask a tutor to examine your solution. The tutor will then ask you some questions to test your understanding. Note that tutors will only be available to assess your solution during the official lab session (Mondays, 9am to 11am, A32).

For all of the exercises on this sheet, you can check your solution by loading your script into GHCi. If you have made a type error it will be detected and reported by GHCi. I recommend reloading your script after adding each definition, so that any errors can be detected immediately.

Ensure that your script type checks before submitting it to a tutor for assessment. If you are unable to successfully complete some of the definitions then comment out those that are incomplete or incorrect. In Haskell, multi-line comments start with `{-` and end with `-}`, while single-line comments begin with `--` and extend to the end of the current line.

1. Fill in the gaps (the question marks ?) in the following definitions to make them type correct. It does not matter what the resulting definitions do provided they are type correct. Note: you should type in and complete each definition one at a time; do not copy and paste from the sheet as the Unicode characters will cause errors in GHCi.

```
e1 :: ?
e1 = [False, True, False]

e2 :: ?
e2 = [[1, 2], [3, 4]]

e3 :: (Char, Bool)
e3 = ?

e4 :: [(Char, Int)]
e4 =  ?

e5 :: ? → ?
e5 x = x ∗ 2

e6 :: Int → Int → Int
e6  ? ? = ?

e7 :: (?, ?) → ?
e7 (x, y) = x

e8 :: (?, [Float])
e8 = (['a', 'b', 'c'], ?)

e9 :: a → (a, a)
e9  ? = ?
```

2. When you submit your solution to a tutor, you will be shown some definitions and asked to state their type signatures. Before submitting, you should practise by adding type signatures to the following definitions: (try to give the most general type possible)

```
onetofive = [1, 2, 3, 4, 5]

cards = ['A', '2', '3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q', 'K']

one = '1'

double x = x + x

swap (x, y) = (y, x)

pair x y = (x, y)

toList (x, y) = [x, y]
```