School of Computer Science, University of Nottingham

G51FUN Functional Programming, Spring 2015

Graham Hutton

Coursework I, Part 4/5

Monday 16th February 2015

Deadline: Monday 2nd March 2015, 11am

This exercise sheet covers material from Lecture 5 (List Comprehensions), and is worth 20% of the first coursework. Each question is worth a quarter of the marks. 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 you should ask a tutor to examine your solution. The tutor will then ask you some questions to test your understanding. You will not receive any marks if you cannot explain your solution. Note that tutors will only be available to assess your solution during the official lab session (Mondays, 9am to 11am, A32).

Ensure that your script type checks before submitting it to a tutor for assessment. If you are unable to complete some of the definitions then comment out those that are incomplete.

1. Using a list comprehension, define a function that selects all the even numbers from a list.

   $evens :: [Int] \to [Int]$

   You may find the library function $even :: Int \to Bool$ helpful.

2. (a) Using a list comprehension, define a function *squares* that takes a natural number n as argument and returns a list of the first n square numbers. For example, *squares* 4 should return $[1, 4, 9, 16]$. You should include a type signature with your definition.

   (b) Define an expression *fiftySquares* that calculates the sum of the first 50 square numbers.

3. (a) Using a list comprehension, define a function that returns a list of all coordinate pairs on an $m \times n$ rectangular grid, where m and n are natural numbers.

   $coords :: Int \to Int \to [(Int, Int)]$

   (b) Using a list comprehension, define a function that returns a list of all coordinates on an $n \times n$ square grid, excluding the diagonal running from $(0, 0)$ to (n, n).

4. **Bonus Exercise**. (Optional, but you will earn additional marks if you complete it.)

   Define a function $clean :: String \to String$ that removes all non-alphabetic characters from a string, and converts all upper-case letters to lower-case letters. For example,

   clean `"Haskell is FUN!!!"`

   should return

   `"haskellisfun"`

   Remember that a string is just a list of characters. You may find it helpful to include the following code **at the top of your file**:

   import $Data.Char$ $(isAlpha, toLower)$

   This will import the functions *isAlpha* and *toLower* from the *Char* library.