School of Computer Science, University of Nottingham

G51FUN Functional Programming, Spring 2012

Neil Sculthorpe

Coursework I, Part 1/5

Thursday 2nd February 2012

Deadline: Friday 10th February 2012, 4pm

This exercise sheet is worth 20% of the first coursework. It is assessed on a simple pass/fail basis: if you complete the sheet you will receive full marks, otherwise no 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.

Note that tutors will only be available to assess your solution during the official lab session (Fridays, 2pm to 4pm, A32).

The most commonly used Haskell compiler is the Glasgow Haskell Compiler (GHC). GHC also provides an interactive interpreter (GHCi), which will be the main tool used in this module. The recommended way to write Haskell programs is to have two windows open: one for a text editor to write your code, and the other for GHCi so that you can regularly load and test your code.

1. Open a text editor of your choice. On the Windows machines in the lab I recommend Notepad++. If you're using your own machine and are familiar with Emacs, then I recommend installing the Emacs Haskell Mode and using that.

2. Type in the following function definition:

   $$double\ x = x + x$$

3. Save your file as

   `Script1.hs`

   The name does not really matter, but it is important that it has a `.hs` file extension (indicating that it is a Haskell source file).

4. Load your file into GHCi. If you're using Windows on the lab machines, then double-clicking on your file will open GHCi and load the file (provided you correctly saved it as a Haskell source file). If you're using a command line interface (e.g. the School's Linux servers), then navigate to the directory containing your file and type:

   ```
   ghci Script1.hs
   ```

   In either case, GHCi should load and you should see something similar to this:

   ```
   [1 of 1] Compiling Main             ( Script1.hs, interpreted )
   Ok, modules loaded: Main.
   *Main>
   ```

5. Test your function on some numbers. For example, type:

   $$double\ 7$$

6. Add a new function to your **source file** by typing (on a new line):

   $$quadruple\ y = double\ (double\ y)$$

   Then save your file.

7. To test the new function, you need to reload the file into GHCi. This can be achieved by giving the command `:reload` (in GHCi). Now test the quadruple function on some numbers.

8. Add the following functions to your source file, and then test them in GHCi:

   $$smallest\ x\ y = \textbf{if}\ x < y\ \textbf{then}\ x\ \textbf{else}\ y$$
   $$largest\ m\ n = \textbf{if}\ m > n\ \textbf{then}\ m\ \textbf{else}\ n$$

   Don't forget to reload the file before testing it, or to save the file before reloading it.

9. Tab characters can cause problems when a file is used by different programs, as some programs interpret tabs as 4 spaces, some as 8 spaces, and some as a single character. Haskell syntax uses layout as an alternative to curly braces and semicolons, which means the width of a tab can affect the meaning of a program. **The best way to avoid problems is not to use tab characters in your source files.** However, if you really want to use the tab key when writing Haskell code, then ensure that your editor either uses tabs of width 8, or automatically converts tabs to spaces. This can be achieved in Notepad++ by going to "Settings/Preferences", selecting the "Language Menu / Tab Settings" tab, and then changing either the default settings or Haskell-specific settings.

   Now add and test the following code, making sure that the l and s after the **where** line up in the same column.

   $$\begin{aligned} &diff\ x\ y = l - s \\ &\qquad \textbf{where}\ l = largest\ x\ y \\ &\qquad\qquad\ \ s = smallest\ x\ y \end{aligned}$$