CSCI 239—Discrete Structures of Computer Science

Lab 1—Introduction to Haskell (Pre-lab)

The pre-lab is a simple exercise in using Haskell.

Objectives:

- to become acquainted with the *ghci* Haskell environment
- to learn to use built-in Haskell programs
- to write a few simple Haskell programs.

Preliminaries:

If you have not already done so, make a *CS239* directory in your Linux home directory. Copy the directory `/usr/people/classes/CS239/labs/Lab01_HaskellIntro` into your *CS239* directory.

Open a terminal window, and type `ghci` at the command prompt. You should see something like the following:

```
GHCi, version 7.0.4: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
Prelude>
```

Using the *ghci* program

Type "`:?`" at the *ghci* prompt. You get a list of the *ghci* system commands, all of which start with the '`:`' character. We will mostly be interested in the first group, the commands available from the command prompt.

Now let's try a few things. If we type something at the prompt that doesn't start with '`:`', the program tries to interpret it as Haskell code. Try entering "`5`".

You should get "`5`" back because "`5`" means 5 in Haskell. Now enter "`5*5`".

Do you get what you expect? You should get "`25`" since "`5*5`" means 5 multiplied by 5 in Haskell. Try entering a few more simple arithmetic expressions using both integer and floating-point numbers. Note cases where the result is different than you expect. How do you do integer division?

Using simple Haskell programs in *ghci*

Now let's see how we can develop programs in Haskell. Use the "`:cd`" command at the *ghci* prompt to change into your *CS239/Lab01* directory. (If you were already in this directory when you ran *ghci*, you can skip this command.) The directory should contain the file *Lab01.SampleCode.lhs*. You can verify this by using the "`:!`" command to run the shell command "`ls`" thus: "`:! ls`". Similarly, ":! pwd" will tell you what directory you'r currently in.

Open the file *Lab01.SampleCode.lhs* in a text editor such as *gedit* so you can view the Haskell code. Next, load the file into *ghci* using the command:

```
:load Lab01.SampleCode.lhs
```

You should see a couple of lines of compiler messages followed by a new *ghci* prompt: `*Main>`.

Notice that the file contains the definition of several Haskell functions. Try the following commands to test one of these functions:

```
square 5
square 4.32
square -4.6
square (-4.6)
```

Why did "`square -4.6`" fail and "`square (-4.6)`" succeed?

Try the *double*, *max2a*, *max2b*, and *lcd* functions with various choices of parameters. Based on what you learned using the *square* function, how should you handle negative parameters with these other functions? Does the *lcd* function work correctly for negative numbers? If not, any ideas on how to correct it?

Writing simple Haskell programs in *ghci*

Write a function called *fourth* that computes the fourth power of its argument by calling *square* twice. Put the function in the same file with the sample programs, and be sure to add comments as to what it does and that you are the author of the function. Use the "`:reload`" command to compile and load your changes into *ghci*. There are at least two different ways to do this, but note that it may be necessary to use parentheses to make it work. Test your function on both integer and floating-point arguments.

Write a function called *eighth* that computes the eighth power of its argument by calling some combination of *square* and/or *fourth*. Again, there are multiple ways to do this, and again, test your function on several arguments.

To exit from *ghci* type "`:quit`" at the prompt.

Be prepared to show your pre-lab work at the beginning of Lab 1.