CSCI 239—Discrete Structures of Computer Science

Lab 1—Introduction to Haskell

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:

Complete the pre-lab exercise for this lab before continuing below.

If you have not already done so, open the file *Lab01.SampleCode.lhs* in a text editor such as *gedit*, and run *ghci* in a terminal, changing to the *Lab01_HaskellIntro*, directory if necessary.

Part 1: Evaluating expressions in Haskell

Consider computing one of the roots of a quadratic equation of the form:

- $ax^2 + bx + c = 0$

One of the roots has the value:

- $\frac{-b+\sqrt{b^2-4ac}}{2a}$

In the text file, put Haskell code to bind the identifiers $a = 1$, $b = -5$, $c = 6$, and r to a Haskell expression for the indicated root. Load or reload the file into *ghci* and check the value of r; it should be 3.0. Now try the same thing for some other values of a, b, and c by changing the values of the variables in the file and reloading it. What happens if $b^2 - 4ac$ is negative?

As you probably discovered, entering expressions by hand is not much fun, particularly when the expressions get complicated. But that's what functions are for! In the next section of the lab, we'll explore how to use functions in Haskell.

Part 2: Writing and using functions in Haskell

For the following sections, put any Haskell code you write in the file you have been using already. Copy the results of interpreting the program as well as the results of any required runs of the program from the terminal window into a separate report document (text or Word). In Linux, you can do this by marking the area in the terminal window and, then, *middle*-clicking at the place in the report window where you want the result to go.

Function 1: A function to evaluate a quadratic polynomial

To begin, enter your function and appropriate comments using the text editor on *Lab01.SampleCode.lhs*. When you are ready to try your function, save *Lab01.SampleCode.lhs* from your editor, and then type "`:reload`" at the *ghci* command line (or "`:load Lab01.SampleCode.lhs`" if you are starting again) to load the current version of the *Lab01.SampleCode* file with your new function. If you get error messages you must fix the file and do "`:reload`" again.

Design a function *quadpoly*, which computes the value of a quadratic polynomial for a particular value of *x*. It should take four parameters representing the values of *a*, *b*, *c*, and *x* in the polynomial $ax^2 + bx + c$. Use the *square* function from the pre-lab as part of your function. The type of the new function is:

```
quadpoly :: Num a => a -> a -> a -> a -> a
```

Try your *quadpoly* function with several sets of values. Use it to evaluate the polynomial $x^2 + 1$ at -2, 0, and 2 by typing "`quadpoly 1.0 0.0 1.0 (-2.0)`", then "`quadpoly 1.0 0.0 1.0 0.0`" and "`quadpoly 1.0 0.0 1.0 2.0`". (Why are there parentheses around (-2.0) in the first call?)

Now define a function *poly1 x* that uses the *quadpoly* function to evaluate the polynomial $3x^2 - 25x + 7$. (Hint: define it with a call to *quadpoly* and the appropriate coefficients.) Evaluate this polynomial for all integer values from −5.0 to 5.0. You should find that the values change sign in this region, indicating the presence of a root. (A *root* of a polynomial is an *x*-value where the polynomial evaluates to zero.) Evaluate the polynomial at the halfway point between the two values where the sign change occurs, and repeat dividing the interval where the sign changes in half and evaluating the polynomial at the halfway point until you have computed the value of *x* to three digits of accuracy. (Note: you don't have to divide the interval exactly in half. You can often make a closer estimate and get another digit in only two or three calls to *poly1*.) Life shouldn't be this tedious; is there an easier way?

Function 2: A function to evaluate a cubic polynomial

Implement a function *cubicpoly* that evaluates the cubic polynomial $ax^3 + bx^2 + cx + d$. Implement a function *cube* that cubes its argument as part of your strategy for *cubicpoly*. Define a function *poly2* that uses the *cubicpoly* function to evaluate the polynomial $x^3 - 5.359375$. Evaluate this polynomial for all integral values from −5.0 to 5.0. Use the same method as for the *poly1* function to find a root. (In this case, you should be able to find an exact root in only a few steps.)

Function 3: A function that uses a user defined data type

Define a Haskell function *triangle* that takes thee parameters *a*, *b*, and *c* and computes whether a triangle with those side lengths is *equilateral* (all three sides are equal), *isoceles* (two sides are equal), or *scalene* (all three sides are different). You will need to use a conditional structure like that in the *lcd* function with guard expressions that have compound conditions; that is, you can combine simple conditions with the *andalso* operator (`&&` in Haskell), the *orelse* operator (|| in Haskell), and/or the *not* function. Your function should return a *String* (same syntax as Java Strings) with the appropriate result.

Test your function with several data sets. (There are five different cases to test, depending on which two or three of the parameters are equal.)