The goal of this assignment is to introduce you to basic Haskell syntax and to help you become familiar with the GHC interpreter (ghci). If you want to install and use GHC on your personal computer or laptop (highly recommended), you can download it from http://www.haskell.org. General information about Haskell is also available on the HaskellWiki (http://www.haskell.org).

Instructions. Provide answers to the following steps and turn in your assignment by the due date. Be sure your name is clearly marked on your assignment. Also, clearly mark the answers to each question (i.e., the question number and letter as appropriate).

1. Start ghci and issue the help command

   Prelude> :help

   Note that you can also use :? for help within ghci. Now issue the command

   Prelude> :info round

   What does this return?

2. Enter each of the following commands into ghci and list what they return and whether the result was what you expected. If the command gives an error, take a guess at what the problem is.

   (a) 5 + 8
   (b) 3 * 5 + 8
   (c) (+) 5 8
   (d) 2^1000
   (e) even 9
   (f) succ 6
   (g) succ (pred 6)
   (h) succ pred 6
   (i) gcd 21 27
   (j) gcd(21, 27)
   (k) [1,2,3,4]
   (l) [1..4]
   (m) [1..4] ++ [5..10]

---

1To install on Ubuntu, from a terminal type sudo apt-get install ghc
2Note that round is a function defined within the Haskell “Standard Prelude” (the default Haskell library), which is documented here (assuming you are using ghc 7.0.4 or higher): http://www.haskell.org/ghc/docs/7.0.4/html/libraries/base-4.3.1.0/Prelude.html
1: \([2,4..8] ++ [9,11..16]\)

(o) \([1, 2, 3.14]\)

(p) \([1, "2"]\)

3. List the types for each valid expression in Question 2 (i.e., those that don’t result in an error). In ghci you can obtain the type for an expression \(e\) by issuing the command

\[
\texttt{Prelude} > \texttt{:type } e
\]

(e.g., \texttt{:type even 9}). The type of an expression is everything after the \(::\) symbol (read as “has type”). Be sure to list the type returned as it is given.

4. Answer the following using ghci:

(a) Define \(zp\) to be the infinite list of positive integers (i.e., \(\mathbb{Z}^+\)). \textit{Hint 1}: Use the \texttt{let} \(v = e\) syntax to assign the result of expression \(e\) to \(v\). \textit{Hint 2}: Use the list enumeration syntax \([n..]\) to create an infinite list starting at number \(n\).

(b) What is the type of \(zp\)?

(c) What happens when you enter

\[
\texttt{Prelude} > zp
\]

at the command prompt? \textit{Hint}: to stop printing hit Ctrl-C.

(d) What is the result of executing

\[
\texttt{Prelude} > \texttt{head zp}
\]

(e) What is the result of executing

\[
\texttt{Prelude} > \texttt{head (tail zp)}
\]

(f) What is the result of executing

\[
\texttt{Prelude} > \texttt{head (drop 100 zp)}
\]

5. An often used function in Haskell (and functional programming in general) is \texttt{filter}.

(a) What is the result of executing

\[
\texttt{Prelude} > \texttt{filter even [1..10]}
\]

(b) What are the types of the two arguments being passed to \texttt{filter} in this call? (Again, to obtain the types use the \texttt{:type} command.)

(c) What is the type of the \texttt{filter} function itself (i.e., \texttt{:type filter})?

(d) In Haskell, we can create “partially defined” functions (we’ll talk more about how this works later). For example, execute the following:

\[
\texttt{Prelude} > \texttt{let evens = filter even}
\]

Now, what is the result of executing:

\[
\texttt{Prelude} > \texttt{evens [1..10]}
\]

(e) What is the type of your new \texttt{evens} function? Make a guess as to why \texttt{evens} has this type.