COMP 3649 Tutorial – Introduction to Haskell

GHCi is the Haskell interpreter built upon the GHC Haskell compiler. It has been installed on the lab PCs and can also be installed at home. This tutorial will familiarize you with the environment.

1. To begin, log in to a lab PC. From the start menu, launch GHCi.

   You should find yourself at the "`ghci>` " prompt. The interpreter is waiting for you to enter an expression[1].

   Enter:

   ```
   ghci> 2 + 3
   ```

   When you hit <*enter*>, GHCi evaluates the expression and prints the result.

   In fact, you can think of GHCi as a programmable calculator (we'll get to the "programmable" part shortly). You enter an expression. GHCi evaluates the expression and prints the result.

2. Enter each of the following expressions in order. Before hitting <*enter*>, try to anticipate each result (some may be errors). Also, explain each result.

   ```
   42

   7 + (-3) * (10 - 1)

   7 + -3 * (10 - 1)

   2^5

   it

   it + 1

   it

   13 / 5

   div 13 5

   mod 13 5

   13 `mod` 5
   ```

   ← note: these are *back* quotes

   ```
   (+) 2 3

   abs -3
   ```

[1] The time spent running the interpreter is called a "session".

```
abs (-3)

x

True

not (False || True)

not False || True

1 > 2

1 == 2

1 /= 2

'a'

"a"

'abc'

"abc"

"abc" ++ "def"

square 5
```

3. What is the effect of pressing the up and down arrow keys at the prompt?

   Can the left and right arrow keys also be used?

4. Quit GHCi by entering the "quit" command:

   ```
   ghci> :q
   ```

   This is short for "`:quit`". GHCi commands begin with a colon, and sometimes may be abbreviated to "`:`*c*", where *c* is the first letter in the full name.

5. Navigate to a working folder of your choice (e.g., create a `COMP 3649\tutorials\intro` folder within your home directory).

   Within the folder, right click on an empty space to pop up the menu. Choose "New → Text Document". Name it "`intro.hs`"[2].

   Right click on this file's icon and open it for editing[3]. Of course, the file is initially empty.

[2] Haskell source files are called "scripts" and end in the "`.hs`" extension (or "`.lhs`" which has a special meaning not covered here).

Type in the following (you are defining a `square` function):

```
square x = x * x
```

Save the file but leave the editor open.

Launch GHCi with "`intro.hs`" as its command-line argument (the precise mechanism for doing so depends on the editor you are using – see the footnote). If the interpreter had already been open, you could also have used the "load" command. Fix any errors before continuing.

Evaluate:

```
ghci> square 5
```

6. Flip back to the editor and change the script's contents to be:

```
square :: Int -> Int
square x = x * x
```

You are adding a declaration of the function's type, which is good practice.

Save the file and flip back to the interpreter. Execute the "reload" command, and confirm that there are no errors.

You can leave both the editor and the interpreter open simultaneously as you work, and simply flip back and forth between the two. This allows you to write some code, then test it, then easily make a few more changes and test them, etc., without having to constantly re-launch the tools.

7. When an expression is entered, GHCi:

   a) checks its syntax
   b) if the syntax is valid, checks its type
   c) if the type is valid, evaluates the expression

[3] You may use a code editor of your choice. On the lab PCs, Notepad++ and VS Code have both been installed. If you choose to use VS Code, downloadable extension support exists, which provides support for Haskell script editing and for running code within a GHCi terminal. This must be enabled within VS Code for each user, the first time they launch VS Code on a particular machine. If using a different editor, such as Notepad++, you may leave your code open within the editor window (remembering to save the code before testing it) while also interacting with GHCi and running the code in a separate terminal window.

GHCi can be told to perform just the first two and then to print the expression's type. Try the following (some of the output may appear strange; this will be explained later).

```
:type square

:t 5

:t square 5

:t 'a'

:t square 'a'

:t not True && True

:t 1 +

:t 1 + 'a'

:t 1 + 2

:t (+) 1 2

:t (+) 1

:t (+)
```

8. At the prompt, use "`:?`" to get a list of GHCi commands. Read the list of available commands and familiarize yourself with the purpose of each.

   The some of the most important and useful are:

   ```
   :load <filename(s)>
   :reload
   :type <expr>
   :info <name>
   :quit
   :?
   ```

   To get more information on GHCi, documentation and other helpful information is accessible via the Help menu as well as on the web (see the URL below).

9. Haskell's "standard prelude" is the set of library definitions that are pre-loaded by default. It includes a `sum` function that, given a list of numbers, produces the sum of all numbers in the list.

For example:

```
ghci> sum [5,10,15]
30
```

Try this out.

One way of defining this function yourself would be as follows:

```
sum []     = 0
sum (x:xs) = x + (sum xs)
```

10. GHC and GHCi are part of the “Haskell Platform”, which is free software you can install on a laptop or home PC. If you want more information about the Haskell, visit the Haskell home page:

    ```
    http://www.haskell.org/
    ```

    This site includes a variety of easy-to-follow Haskell tutorials and other resources (check them out)!