


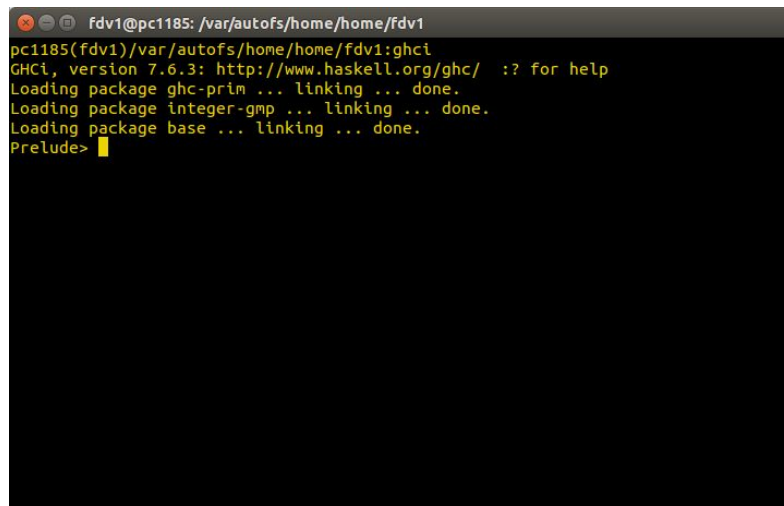
## Worksheet 0: *Getting started with GHCi*

---

Template file:	Worksheet0.hs
Lab :	Friday 22th January
NOT ASSESSED.	
Topics:	gchi (Glasgow Haskell compiler interpreter) commands, evaluation and typing of expres

---

1. we presume that you use Haskell in a linux environment either on your lap-top, PC or remotely logged in to the Computer Lab. <https://campus.cs.le.ac.uk/labsupport/using1>
2. Boot in Linux. Run Firefox and download the file Worksheet0.hs from the web page:  
<https://campus.cs.le.ac.uk/teaching/resources/C02008/code/Worksheet0.hs/>
3. Use the terminal icon (by default on the top panel in Gnome): . Or try “Ctrl Alt t”.
4. Go to the directory that contains Worksheet0.hs. Type `ghci` at the command line, hit return, and the GHC interpreter will start up:



```
fdv1@pc1185: /var/autofs/home/home/fdv1
pc1185(fdv1)/var/autofs/home/home/fdv1:ghci
GHCi, version 7.6.3: 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> 
```

5. The GHCi interpreter evaluates expressions typed at the prompt as demonstrated in class.

Type the following expressions and hit return each time. What are the values of the following expressions?

```
Prelude> 6*7
Prelude> 5-3*4
Prelude> 5-3*4 == (5-3)*4
Prelude> 2 ^ 3
Prelude> 10-2/2
Prelude> 100 >= 12*12
Prelude> "David"<"Diane"
Prelude> True || False
Prelude> True && False
Prelude> div 10 3
Prelude> mod 10 3
Prelude> 10 `div` 3
```

Note that `div`` surrounded by backquotes is written between its two arguments. This is the infix version of the function `div`.

```
Prelude> (*) 6 7
```

Note that `(*)` surrounded by parentheses is written before its two arguments. This is the prefix version of `*`.

```
Prelude> ("John", 2345289)
```

This is a pair of a name and a number.

```
Prelude> 2:5:1:[]
```

A shorthand for `2:5:1:[]` is `[2,5,1]`. This is the list that contains 2, 5 and 1. Lists in Hugs correspond to what were called *Sequences* in CO1004. A list is built up in a unique way from the empty list `[]` and the constructor operator `:`.

```
Prelude> head [2,5,1]
Prelude> tail [2,5,1]
Prelude> null [2,5,1]
Prelude> null []
Prelude> [2,5,1]++[7,1,4]
```

```
Prelude> "One is followed by"++" two"
```

In Haskell a string is a list of characters. Thus, `"Hallo"` is just a convenient additional syntax for the particular list `['H','a','l','l','o']`.

```
Prelude> reverse "Madam'I"
```

```
Prelude> putStr "One Two Three"
```

```
Prelude> putStr "One\nTwo\nThree"
```

6. GHCi commands begin with a colon ‘:’. A summary of the main commands follows:

<code>:load Worksheet0</code>	Load the Haskell file <code>Worksheet0.hs</code> . The file extension <code>.hs</code> does not have to be included in the filename.
<code>:reload</code>	Repeat the last load command
<code>:quit</code>	Exit GHCi
<code>:?</code>	Give a list of the GHCi commands.
<code>:type exp</code>	Give the type of the expression <code>exp</code> .

All the ‘:’ commands can be shortened to their initial letter. For instance, type `:l Worksheet0` instead of `:load Worksheet0`. Try them all out. Experiment!

7. We can evaluate expressions which use definitions from a file that carry the extension `.hs`. Using XEmacs read the definitions of the constants and function from the file `Worksheet0.hs`. Note that the symbol ‘-’ begins a comment. Load the file `Worksheet0.hs` into GHCi. What does the prompt look like now? Evaluate the constants `myint`, `myfloat`, `mychar`, `mystring`, `less` and the function `cube` applied to the argument 2.
8. Complete the definitions of `plus` and `allEqual` in the file `Worksheet0.hs` (remove the comments before the type declarations). Reload the file and check if your definitions seem to be correct by testing on some examples (everytime you make changes in the file you have to reload it).
9. Try to evaluate the following expressions and make sure you understand the error messages.

```
Worksheet0> 2*(6+8
Worksheet0> 8/(7-7)
Worksheet0> plus True 2
Worksheet0> myint+myfloat
Worksheet0> mystring++myint
Worksheet0> mystring++mychar
```

10. The last three expressions in the previous exercise can be evaluated if we use special functions to convert from one type to another. Find out the type of the following expressions using the command `:type` and then evaluate them.

```
myint+(floor myfloat)
(fromIntegral myint)+myfloat
```

```
mystring++(show myint)
mystring++[mychar]
```

11. Define a function `message` that given a real number `x` returns the message "The number is `x`". For instance, `message 30.22` should return "The number is 30.22".
12. Write a function `blankVowel` that given a character `x` returns a blank when `x` is a vowel (ie.  $x \in \{a, e, i, o, u, A, \dots\}$ ) and other wise returns `x`.
13. (from Hutton's book) The script below contains three syntactic errors. Correct them and check that your script works properly using GHCi.

```
N = a 'div' length xs
  where
    a = 10
    as = [1,2,3,4,5]
```