CA2320 Computability and Complexity

Functional Programming Lab 1: Getting Started

Aim

The aim of this practical is to show you how to use the Glasgow Haskell Compiler (GHC) system. This will involve becoming familiar with the GHC's interpreter (GHCi), entering expressions into GHCi, creating and loading your own scripts into GHCi.

1. Starting Hugs

To start GHCi log in to Windows and click on Programs → Programming & Development → Glasgow Haskell Compiler 7.0.4 → GHCi in the START menu. This should start the Glasgow Haskell Compiler interpreter and you should see something like:

```
GHCi, version 6.12.3: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
Loading package ffi-1.0 ... linking ... done.
Prelude>
```

2. A GHCi Session

You can now enter some expressions at the prompt and the Haskell interpreter will evaluate them. Try the following examples. Many of them use functions provided in the standard prelude. Are the answers what you expected?

```
3
3.0
'3'
"3"
3+4
20 `div` 5
20 `mod` 5
```

Make sure you used the correct apostrophe in the previous two expressions.

```
20 / 5
20.0 / 5.0
2 ^ 5
sqrt 2.0
sqrt 5*5
sqrt 15.0 >= 3.5
sin (pi/2.0)
```

```
6 < 4
3>4 && 4>3
3>4 || 4>3
not (3>4 && 4>3)
"Hello World!\n"
putStr "Hello World!\n"
“abc" ++ "def"
length “This sentence no verb.”
words “The quick brown fox jumped over the lazy dog”
“The length of \”this sentence\” is :” ++
     show (length “this sentence”)
```

3. Quiting a GHCi session

To quit a GHCi session type in `:quit` ot `:q`.

4. Creating a Haskell Script

To create your own script, open an editor, EditPlus is loaded on the Windows lab machines, and type in the following:

```
-- CA320 Computability and Complexity
--
-- <your name here>

cube :: Int -> Int
cube x  =  x * x * x

edge, volume :: Int
edge = 3
volume = cube edge

surfaceArea :: Float -> Float
surfaceArea r = 4.0 * pi * r^2
```

Save the file with an appropriate name e.g. exercise1.hs. Don’t forget that the suffix must be `.hs` to indicate to GHCi that it is a Haskell script. To try out the script that you have just created, it must be loaded into GHCi. You can load your script by typing the command `:load exercise1.hs` or `:l exercise1.hs`. If you haven’t made any typing errors, GHCi should successfully load your script. If you have made some typing mistakes, GHCi will give you an error message while attempting to load the script. The error message will include the line number at which GHCi could not continue. After you have made the corrections, you can tell GHCi to re-load by typing the command `:reload`, or just `:r`. When you have successfully loaded your script you should experiment by getting GHCi to evaluate a few expressions using the definitions in your script.