Week 6 Exercises

FIT2102 Programming Paradigms

Requirements

Complete all the exercises as per the instructions found in the code files. All tests must pass. There must be no compilation warnings or errors.

Recommended order: 1. `IntPair.hs` 2. `BinTree.hs` 3. `Examples.hs`

Setting up Haskell

Install GHCup

We will use GHCup to install and manage our tooling.

Start by installing GHCup using the official installation instructions.

- Windows Instructions
- Non-Windows Instructions

Windows You may use Chocolatey to install haskell if you wish via `haskell-dev`.

We will use PowerShell and the provided GHCup script for installation.

In the installation process, select the following options:

```
Chocolatey was detected on your system. It is capable of installing the Haskell toolchain as
If you want to rather use that instead of ghcup, abort the installation and run the followin
elevated command prompt:
    choco install haskell-dev
    refreshenv
```

```
Continue with GHCup installation?
[C] Continue  [A] Abort  [?] Help (default is "C"): C



Picked C:\ as default Install prefix!
Welcome to Haskell!

This script can download and install the following programs:
  * ghcup - The Haskell toolchain installer
  * ghc   - The Glasgow Haskell Compiler
  * msys2 - A linux-style toolchain environment required for many operations
  * cabal - The Cabal build tool for managing Haskell software
  * stack - (optional) A cross-platform program for developing Haskell projects
  * hls   - (optional) A language server for developers to integrate with their editor/IDE

Please note that ANTIVIRUS may interfere with the installation. If you experience problems,
disabling it temporarily.

Where to install to (this should be a short Path, preferably a Drive like 'C:\')?
If you accept this path, binaries will be installed into 'C:\ghcup\bin' and msys2 into 'C:\g
Press enter to accept the default [C:\]: (Enter) or choose your own path. This should be one



Specify Cabal directory (this is where haskell packages end up)
Press enter to accept the default [C:\\cabal]: (Enter) or choose your own path. This should



Install HLS
Do you want to install the haskell-language-server (HLS) for development purposes as
well?
[Y] Yes  [N] No  [A] Abort  [?] Help (default is "N"): Y



Install stack
Do you want to install stack as well?
[Y] Yes  [N] No  [A] Abort  [?] Help (default is "N"): Y



Install MSys2
Do you want GHCup to install a default MSys2 toolchain (recommended)?
[Y] Yes  [N] No  [?] Help (default is "Y"): Y
```

Not Windows In the installation process, select the following options:

```
Do you want ghcup to automatically add the required PATH variable to "..."?

[P] Yes, prepend  [A] Yes, append  [N] No  [?] Help (default is "P").

A

Do you want to install haskell-language-server (HLS)?
HLS is a language-server that provides IDE-like functionality
and can integrate with different editors, such as Vim, Emacs, VS Code, Atom, ...
Also see https://haskell-language-server.readthedocs.io/en/stable/

[Y] Yes  [N] No  [?] Help (default is "N").

N

Do you want to enable better integration of stack with GHCup?
This means that stack won't install its own GHC versions, but uses GHCup's.
For more information see:
  https://docs.haskellstack.org/en/stable/yaml_configuration/#ghc-installation-customisation
If you want to keep stacks vanilla behavior, answer 'No'.

[Y] Yes  [N] No  [?] Help (default is "Y").

Y
```

Run

```
$ ghcup --version
```

to check that the installation has succeeded.

Install tools using GHCup

The tools should be installed automatically, but make sure they are the correct versions.

The versions installed should be as follows:

- Stack: 3.1.1 (recommended)
- HLS: 2.7.0.0 (recommended)

If they are different, install the correct versions.

You may have to restart your terminal for PATH changes to take place.

- Windows Instructions
- Non-Windows Instructions

Windows Run

```
$ ghcup list
```

and check that Stack, HLS, and cabal have been installed with the correct versions.

Run

```
$ ghcup install <tool> <version>
```

to install any missing versions. For example,

```
$ ghcup install hls 2.7.0.0
```

Not Windows Run

```
$ ghcup tui
```

and check that Stack, HLS, and cabal have been installed with the correct versions.

Follow the instructions of the GUI to install any missing versions.

Test

Open the code bundle and ensure stack works by running

```
$ stack ghc -- --version
The Glorious Glasgow Haskell Compilation System, version 9.2.8
```

This may download any missing dependencies.

Windows Issues If you get an error like this

```
Error: [S-7282]
       Stack failed to execute the build plan.

       While executing the build plan, Stack encountered the following errors:

       <stderr>: commitAndReleaseBuffer: invalid argument (invalid character)
```

Go to Settings → Time and Language → Language and Region → Administrative language Settings → Change System Locale… → and check Use Unicode UTF-8. You might need to restart your computer.

Essential VSCode extensions

Haskell - Must have - Haskell language support including language server, syntax highlighting, and linting

Haskell Syntax Highlighting - Useful - Adds syntax highlighting for Haskell associated languages - Adds automatic indentation

Optional VSCode Extensions

These may take a while to setup, so it is worth getting started on your applied session work!

Ormolu - Very Useful - Haskell formatter - Requires installing ormolu separately - This can be done via `stack install ormolu`

hlint - Useful - Additional linting options and configuration - Requires building hlint separately - This can be done via `stack build hlint`

Using the code bundle

Once you have downloaded and extracted the code, set it up by running:

```
$ stack setup
```

- You should only have to do this once throughout the semester

Build your code files by executing

```
$ stack build
```

To run, debug, or test the code, enter Haskell's interactive environment, GHCi[1]:

```
$ stack ghci
```

This will be the only way to test your code this week, so make sure you are comfortable with this terminal

It should load all the source files in the current project. If you modify your code and want to test it, you need to `:load/:l` or `:reload/:r` your code, the former will load the file you indicate, while the latter will reload the whole project:

```
*Main Lib> :l src/BinTree.hs
[1 of 1] Compiling BinTree           ( src/BinTree.hs, interpreted )
Ok, modules loaded: BinTree
*BinTree> :r
[1 of 1] Compiling BinTree           ( src/BinTree.hs, interpreted )
Ok, modules loaded: BinTree.
```

The interactive environment gives you a number of tools to explore the code, typing `:type/:t` will give you the type of a function; to have more information, you can use `:info/:i`.

```
*Maybes> :t mapTree
mapTree :: (Int -> Int) -> BinTree -> BinTree
*Maybes> :i mapTree
mapTree :: (Int -> Int) -> BinTree -> BinTree
    -- Defined at src/BinTree.hs:82:1
```

[1]GHCi is a REPL (Read-Eval-Print-Loop), which means that whatever code you type in it is interpreted as if it were source code; loading a file is equivalent to copy/pasting it.

Using GHCi makes debugging faster as each time you reload your file(s) it will inform you of any syntax or type error.

You can also type code directly within GHCi and use its results either by copy/pasting into your code, or using `it` to recall the last results.

```
Prelude> map (+1) [1..10]
[2,3,4,5,6,7,8,9,10,11]
Prelude> map (*2) it
[4,6,8,10,12,14,16,18,20,22]
```

Example of testing your code

Running `stack ghci` should open up the interactive GHCi console.

```
stack ghci
```

```
ghci> :l src/BinTree.hs
[1 of 1] Compiling BinTree          ( src/BinTree.hs, interpreted )
Ok, one module loaded.
```

The code has successfully compiled. We can now test a function, e.g., `size`. However, we have not completed this exercise, so it returns an undefined.

```
ghci> size tree
*** Exception: Prelude.undefined
CallStack (from HasCallStack):
  error, called at libraries/base/GHC/Err.hs:74:14 in base:GHC.Err
  undefined, called at src/BinTree.hs:34:8 in main:BinTree
```

After completing the exercise

```
ghci> :r
[1 of 1] Compiling BinTree          ( src/BinTree.hs, interpreted )
Ok, one module loaded.
ghci> size tree
4
```

This matches the expected value documented in BinTree, so our answers is most likely correct.