Assignment 0: Setting Up Haskell

CSC324H1, Fall 2025

Due Date: Sept 8, 2025 (11:59 pm)

Overview

In this assignment, you will:

- Install the Haskell toolchain on your computer;
- Familiarize yourself with basic Haskell syntax.

Instructions

For the first part, submit a screenshot of the terminal commands' outputs as demonstrated in the handout.

For the second part, a starter project is provided to you. You should complete the functions as required later in the handout and marked with `error "TODO"` in the code. Run the tests provided to check your work. Note that your code will be graded based on additional hidden tests.

Please note that for all assignments:

- It is recommended to only submit all of the required files as specified on MarkUs.
- You are responsible for making sure that your code has no syntax errors or compile errors. If your file cannot be imported by another file, you may receive a grade of zero.
- If you're not intending to complete one of the functions, do *not* remove the function signature. If you are including a partial solution, make sure it doesn't cause a compile error. If your partial solution causes compile errors, it's better to comment out your solution (but not the signature).
- Do not modify the `module (...) where` lines of your code. These lines are crucial for your code to pass the tests.

Part 1: Installing Haskell (40 pts)

We will be using Haskell and its toolchain throughout the term. Haskell is an industrial-strength functional programming language full of interesting language features.

Your Task

Your task is to demonstrate that you have access to the basic Haskell development tools. The following is a list of tools you need, along with the version number we suggest:

- `GHC`: 9.10.2
- `Cabal`: 3.12.1.0
- `HLS` (optional): 2.11.0.0

For help with local installation, see Installation Guide in the appendix. If you cannot get the exact version of the tools, make sure you have *GHC version* $\geq$ *9.0.0* and *cabal version* $\geq$ *3.0.0.0*.

To ensure you have downloaded the correct version of the tools, run the following commands in your terminal, and upload a screenshot (40 pts) of the output message.

```
$ ghc --version
The Glorious Glasgow Haskell Compilation System, version 9.10.2
$ cabal --version
cabal-install version 3.12.1.0
```

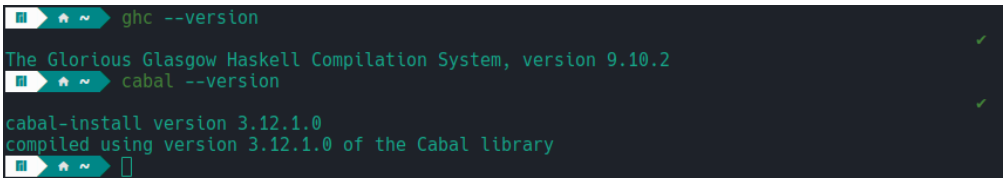

Figure 1: Expected output from terminal commands.

If you are using the teaching lab, the tools are already installed on the school machines. There is no need for manual installation. You can send a screenshot of the lab machine terminal output directly.

Part 2: Basic Haskell (60 pts)

In this part, you will write some simple Haskell programs to get familiar with the Haskell syntax.

You can learn about how to run and test a Haskell Cabal project in the appendix Understanding Haskell Projects.

The starter code is provided in `Basics.hs`.

(a) `power5` (20 pts)

Implement the function `power5 :: Int -> Int`. The function takes an `Int` and returns the number raised to the power of 5.

(b) `ageDiscount` (20 pts)

Implement the function `ageDiscount :: Int -> Float -> Float`. The function takes an age (`Int`) and a base price (`Float`). It returns the base price ×0.8 if the age is less than 18 or greater than or equal to 60. Otherwise, return the base price ×1 .

You might find the `if ... then ... else ...` construct or the *guard* syntax useful.

(c) `squareTheDiff` (20 pts)

Implement the function `squareTheDiff`. The function takes two `Int`s, `x` and `y`, and returns the square of their difference: $(x-y)^2$.

This is a good opportunity to use the `let ... in ...` syntax.

Appendix A: Installation Guide

We provide some standard ways to install Haskell tools on your computer, with a focus on setting up using *GHCup*. You can find the official guide at:

https://www.haskell.org/ghcup/

Alternative installation methods are listed on the Haskell website.
It is your responsibility to ensure compatibility of tools and libraries.

(a) Linux/MacOS System

You can find the install command at GHCup's installation page. Copy and paste the command into your terminal to run the installation script (as *non-root/non-admin user*):

```
$ curl --proto '=https' --tlsv1.2 -sSf https://get-ghcup.haskell.org | sh
```

GHCup will ask you a few questions regarding customizing installation. Make sure to:

- Add the Haskell toolchain to your PATH.
- Install HLS if you want better integration with your IDE of choice.

After the installation script finishes, you should refresh your terminal session and open GHCup:

```
# After re-running your shell startup script (like .bashrc), execute:
$ ghcup tui
```

This should bring up a text-based UI in your terminal, where you can use the arrow keys to highlight, install, and select different versions of tools.

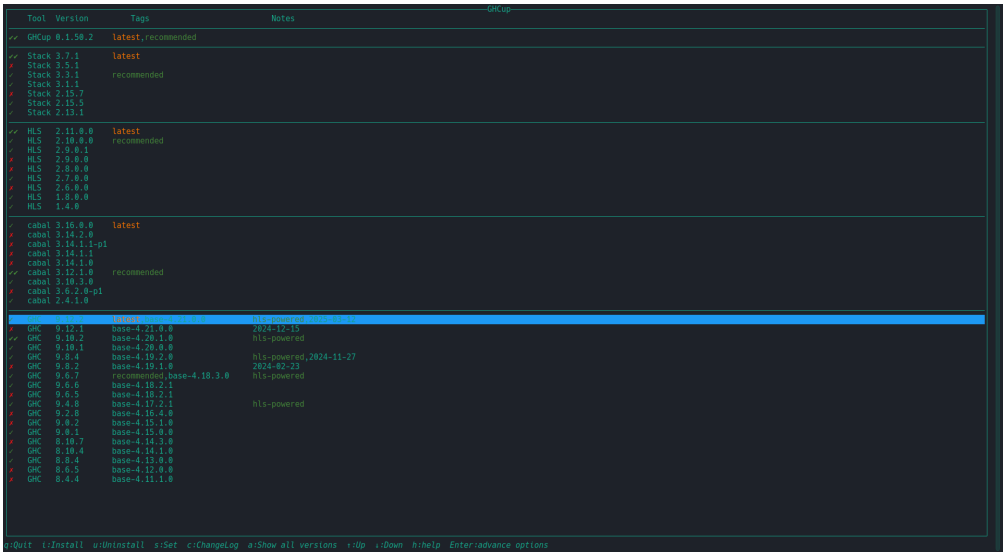

Figure 2: What you should see after running `ghcup tui`, if your installation went correctly.

(b) Windows with WSL

If you are on Windows, we recommend using Windows Subsystem for Linux (WSL).
Start by starting a PowerShell session, and install a distribution (distro) of Linux:

```
> wsl -l -o                    # List all available distributions to install
...
> wsl --install Ubuntu         # We chose Ubuntu
> wsl --set-default Ubuntu     # Always run WSL with Ubuntu
```

The installation will ask you to set up a username and password for WSL. After that, you will be brought into the terminal of the Linux subsystem.
As noted on GHCup's installation page, you will need some basic tools to install GHCup:

```
$ sudo apt-get update
$ sudo apt-get install build-essential curl libffi-dev libffi8ubuntu1 \
                       libgmp-dev libgmp10 libncurses-dev
```

After installing the dependencies, you can follow the installation guide for Linux/MacOS System to finish installing GHCup.

(c) Windows Natively

It is possible to install GHCup directly on Windows, although it might take more work.
You can find the install command on GHCup's installation page. There is also a YouTube video explaining the installation on the website.

Appendix B: Understanding Haskell Projects

All of your assignments will be distributed and collected as *Cabal* projects. This part of the handout will help you become familiar with Cabal projects.

(a) Building a Haskell Project

To build a Cabal project, navigate to the project directory `.../project-dir` so that it contains the Cabal file `project-name.cabal` (`a0.cabal` for this assignment), and run the command `cabal build`:

```
$ cd path/to/project-dir
.../project-dir $ ls
... project-name.cabal ...
.../project-dir $ cabal build
```

(b) Running an Executable in a Haskell Project

If the `project-name.cabal` file contains a line of the form: `executable app-name`, the project contains an executable. You can run the executable with the command `cabal run`:

```
.../project-dir $ cabal run
```

(c) Running Test Suites in a Haskell Project

If the `project-name.cabal` file contains lines of the form: `test-suite test-name`, the project contains a test suite named `test-name`. You can run the tests with the command `cabal test test-name`:

```
.../project-dir $ cabal test test-name
```

A single Cabal project can contain many test suites. Adjust the argument `test-name` to run different suites.

(d) Running a GHCi Session in a Haskell Project

Command `cabal repl` opens an interactive GHCi session for the compilation target within the project and loads all of the modules of the target into GHCi. This is useful for testing your code interactively. You may need to first import the modules of the functions you want to call in the GHCi session.

```
.../project-dir $ cabal repl
# Build outputs appear here...
...
Ok, one module loaded. # This could be more than one
ghci>                  # Call your functions here in the GHCi session
                       # You may need to execute 'import ...' first
```