Assignment 1

COMP 524 - Spring 2019

Due: Tuesday, January 23rd 2018, 11:59 PM EDT on Gradescope

Overview

This assignment intends to help you get familiar with the Haskell interpreter (GHCi) and to introduce you to Haskell's syntax and programming model. You will do this through the virtual machine (VM) and submission infrastructure that you tested with Assignment 0.

GitHub and Git Setup

First accept Assignment 1 on GitHub classroom via this link: https://classroom.github.com/a/bRg2ljs6. Use the same GitHub account as you did with Assignment 0. After that, just like with Assignment 0:

1. Open up the directory where you setup your VM with Vagrant then run `vagrant up` and `vagrant ssh` to get back to the console inside the VM that you have used before
2. Clone the URL `https://github.com/comp524-s19/assignment-1-[YOUR GITHUB USERNAME].git`, replacing the `[YOUR GITHUB USERNAME]` part of that appropriately
3. Move to the directory git created in the VM (it should be called `assignment-1-[YOUR GITHUB USERNAME]`)

Congratulations! You are now ready to start coding in Haskell.

Program Specification

Create a file named `main.hs` on your virtual machine containing your implementation of a Haskell function named `splitit`. It is to take a list of any type, split it in half, and return the two halves in a new list. If there are an odd number of elements, the first half is to be the smaller one. This implies that if there is only one element, the first list of the two should be empty. If the input is the empty list, the output list should contain a pair of empty lists. The input list can contain any type of element, including numbers, characters, strings, lists, etc.

To test your function, run `ghci main.hs` from your VM console. This opens the Haskell interpreter and loads the code you have in `main.hs`. If you ever need to leave the interpreter, simply use the key combination `Ctrl+D` to return to your normal prompt. Inside the interpreter, you can dynamically write and run Haskell code. To test a function, simply call it by typing the name, any parameters, and press enter. In our case, that would look something like `splitit [1,2,3,4]` which should then return `[[1,2],[3,4]]`. Here are a few other test cases to try:

Input Parameter	Result
`[]`	`[[],[]]`
`[1]`	`[[],[1]]`
`[1,2]`	`[[1],[2]]`
`[1,2,3]`	`[[1],[2,3]]`
`"Hello World!"`	`["Hello ", "World!"]`

Submission and Grading

Commit and push `main.hs` with git to GitHub, similarly to what you did for Assignment 0. (The git cheatsheet posted on the website is really excellent reference if this step is tripping you up.) Once your code is up on GitHub, go to https://www.gradescope.com and find "Assignment 1" under the COMP 524 class. You can create a submission by selecting your private repository `assignment-1-[YOUR GITHUB USERNAME]` from the Repository selector and `master` from the branch selector. After fetching your code, the autograder should run with feedback shortly. You may submit as many times as you would like up until the deadline. Grading is fully automated and will include both the test cases mentioned in this document and several additional ones.

When you finish the assignment, we suggest running `exit` to leave the VM console and running `vagrant halt` to shut down your VM and free up some computaitonal resources on your machine.

Some Tips

Nothing in this section is essential for the assignment, but might be fun to toy around with if you would like to optimize your development flow.

1. You can open multiple shells on your virtual machine by running `vagrant ssh` from multiple terminals on your local machine. This might be helpful if you want to have `nano` open in one window and `ghci` in another.
2. If you get frustrated with the lack of syntax highlighting in `nano` and are up for a challenge, `vim` is also installed on the VM. It can be quite a handfull to learn, but many developers enjoy its depth of features. There are many online guides on how to use `vim`, but the three key things you need to know are that the `i` key enables text entry, `Esc :w` saves your file, and `Esc :q!` exits `vim`.