CPS506 - Comparative Programming Languages - Fall 2017

Assignment 3 - Haskell

Frequently Asked Questions
Course Management Form
Resources
Assignments
Tweet
Follow @RyersonCPS506

This assignment is a relatively simple program to capture various aspects of programming languages. This version is in Haskell.

The application is a simple Snakes and Ladders game described here. In addition to the specifications there, the following Haskell-specific parameters will apply:

1. Do `mkdir assign3` to create your package. Save this `.gitignore` file in that directory.
2. Your `Main` module must have a `readFrom` function that accepts a string.
3. Your `Main` module must have a implementation of the `Show` class for your Game type that formats a game as a string.
4. When your `Main` program is run, it must read commands from standard input, passing each line to the `readFrom` function. At the end of the input, it must print the state of the board on standard output. To effect this, make sure your module is called `Main` and include the following function definition:

```
main = do
     input <- getContents
     putStr $ show $ readFrom input
```

5. Your unit tests should be implemented with HUnit (`cabal install HUnit`). The collection of tests should be named `tests` so they can all be evaluated with `runTestTT tests` from a `ghci` shell. Here is a starter test file to copy into the end of your program; you will need to put `import Test.HUnit` at the beginning of your module.
6. Put your ownership information (see the assignment page) in the `assign3/README.md` file.
7. The marker should be able to run your program by entering the following code:

```
$ ghci assign3.hs
ghic> readFrom "board 3 4\nplayers 2\nturns 5"
```

or

```
ghc -o assign3 assign3.hs
./assign3
board 3 4
players 2
turns 5
^D
```

You should do your assignment in the your Git `CPS506` repository in a folder called `assign3`. Every time you have completed a part of the assignment, you should commit it to the repository. You shouldn't wait until everything is complete to do this, it's better to check in regularly. Remember to do `git status`, `git commit`, and `git commit` from somewhere within the repository periodically to make sure you're commiting all of your code. Also remember to **not** add binary files or other files that can be generated from the source. `mix` automatically creates a `.gitignore` file that excludes commonly created files that should be excluded; add to that file if you notice any undesirable files being staged for commiting to the repository. You can so a `git add .` as many times as you want, but you only have to do it once each time there are new files to be included in the repository. In a terminal/command window simply change to the working directory and check-in, for example:

```
cd gitlab/cps506
git commit -m "finished code and tests for snakes and ladders"
git push
```

Dave Mason