Coverage & Logistics

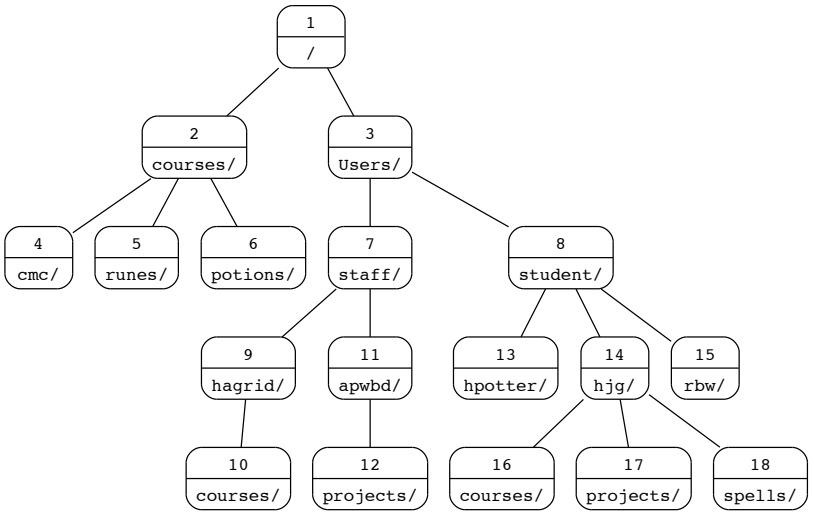
This homework covers material through the first three chapters (with an emphasis on Chapters 1-2) of *Haskell: The Craft of Functional Programming* (HCFP), as well as some Unix basics.

This homework is officially due in class on **Thursday, January 26**. However, it comes with an automatic extension: anything submitted to the CIS 252 bin near CST 4-226 by **noon on Friday, January 27** will be accepted as being on time.

You should work solo on this assignment.

Part 1: Unix Exercises

Consider the following (partial) file system hierarchy:



Here are a couple comments regarding how to read this diagram:

- Each node represents a Unix directory, which may contain additional files that are not included in the diagram.
- Each node contains a numeric label at the top, in addition to the directory's name: for example, the root node (`"/"`) has the numeric label 1. These numeric labels are intended merely as identification for the questions that follow, since (for example) there are three different directories with the name `"courses"`.

Suppose that Hermione's **home directory** is directory #14 and that her **current working directory** is directory #5.

1. What is the **absolute pathname** for Hermione's home directory?
2. If Hermione executed the Unix command `pwd`, what would the system's response be?
3. If Hermione executed the Unix command `cd ..` and then executed the command `pwd`, what would the system's response be?
4. If Hermione executed the Unix command `cd ~/courses` and then executed the command `pwd`, what would the system's response be?
5. For each of the following tasks, give the Unix command (or series of commands) that Hermione would need to complete the task; in many cases, there will be more than one correct answer. You should assume that each task is completed independently of the others (i.e., changes made in one task do not affect other tasks), that her home directory is directory #14, and that her **current working directory** is directory #5.

   As an example, I've completed the first one ($\star$) for you.

   $\star$ Get a long listing (i.e., includes file permissions and creation dates) of directory #4
   SAMPLE ANSWER: `ls -l ../cmc`

   (a) Make a subdirectory called `new` in directory #14
   (b) Move the file `results.pdf` from directory #18 to directory #16
   (c) Copy the file `polyjuice` from directory #6 to directory #17, but give it the name `mission`
   (d) Change her working directory to directory #12
   (e) List **all files** (including those that begin with a period) in her home directory
   (f) Create a subdirectory in directory #16 called `hws` and then place a copy of the file `hw2.hs` (located in directory #5) into that new subdirectory

Part 2: Haskell Exercises

Please note: The solution for each problems **should be written in a single line (two lines, if you count the type declaration) of Haskell.**

General advice: Work on one problem at a time: get it working before moving on to the next problem. (Writing all your code at once will likely result in a series of errors and a lot of frustration on your part. It is usually **much easier** to work in small increments.)

6. Write a **one-line Haskell function**

   ```
   between :: Int -> Int -> Int -> Bool
   ```

   such that `between m y z` returns `True` exactly when `m` is (strictly) between `y` and `z` (i.e., y is less than m and m is less than z); the function should return `False` in all other cases.

   For example, `between 10 3 15` should return `True`, but `between 3 7 6`, `between 8 5 7`, and `between 10 10 15` should all return `False`.

7. The ***exclusive or*** of two boolean values is calculated as follows:

   - When *exactly one of the two values* is `True`, their exclusive-or is `True`.
   - When *both values* are `True` (or both are `False`), their exclusive-or is `False`.

   Write a **one-line Haskell function**

   ```
   xor :: Bool -> Bool -> Bool
   ```

   such that `xor e1 e2` calculates the exclusive-or of `e1` and `e2`.

   For example, `xor True False` and `xor False True` should both return `True`, where `xor True True` and `xor False False` should return `False`.

8. You're probably familiar with the Fahrenheit and Celsius temperature scales, but they are not the only temperature scales to ever be used. The *Delisle* temperature scale was invented by Joseph-Nicolas Delisle in 1738.

   The relationship between a Fahrenheit temperature f and the corresponding Delisle temperature d is given as follows:

   $$f = 212.0 - \frac{6}{5}d$$

   For example, 10° Delisle is 200° Fahrenheit; −5° Delisle is 218° Fahrenheit. (Yes, in the Delisle scale, "warmer" temperatures have smaller numbers associated with them.)

   (a) Write a **one-line Haskell function**

   ```
   convertDtoF :: Float -> Float
   ```

   such that `convertDtoF temp` returns the Fahrenheit equivalent of the Delisle temperature `temp`. For example, `convertDtoF 10` returns `200.0`.

   (b) Write a **one-line Haskell function**

   ```
   convertFtoD :: Float -> Float
   ```

   such that `convertFtoD temp` returns the Delisle equivalent of the Fahrenheit temperature `temp`. For example, `convertFtoD 200` returns `10.0`.

What to turn in

You will need to submit the following (please staple your pages!):

- *The disclosure cover sheet*
- *Part I: Unix questions* Turn in written (or typed) answers to the Unix questions.
- *Part II: Haskell exercises* Turn in hard copies of (1) your source code and (2) a transcript demonstrating convincingly that your code is correct.

  To generate the transcript:

  – Include **all** of the tests that appear in the hw1tests file. (You may include additional tests if you'd like, but these tests are necessary.)

  – Generate your final transcript **only after you have gotten all of your code to work**. Take pity on the graders, and make sure that your solutions will be easy to grade/verify. (If they have to work too hard to find your answers, you will lose points.)