Assignment 3

Due on **February 3rd, end of the day**. Please follow the submission instructions in the "notes for all labs" on Moodle. You are required to do all of this assignment, but the bolded parts are what you should submit.

- Read the syllabus, linked on Moodle. You are responsible for the contents of the syllabus.
- Read the "Notes for All Labs" document on Moodle. All assignments must be submitted as specified there.
- Re-read Chapter 3 of your text. This assignment mostly follows along with the material covered in that chapter. You may want to review Chapters 1 and 2 for help with syntax.
- All of your code should be in a file called your_id_330_lab3.hs, where your_id is your Earlham user ID.
- If you are still unable to run Haskell code, you can write your code in a normal text editor and submit it. Please indicate if this is what you have done so that I will no mark you off for minor typo-induced syntax errors and the like.
- If you haven't managed to get set up with a Haskell-aware editor by now, do it before you work on this assignment. It will make it much easier to read your code, and may include features that help you format your code in a readable and correct fashion.

Written Assignment 3.1: (2 points)

In a sentence or two, describe the difference between `Int` and `Integer` in Haskell.

Written Assignment 3.2: (2 points)

In a few sentences, describe the difference between guards and general conditional expressions (p. 51), and when it is appropriate to use each.

Written Assignment 3.3: (4 points)

In a few sentences, describe what "hiding" means, and try to come up with an example of where you might want to use it. If you're rusty on how modules work, check back in Chapter 2.

Written Assignment 3.4: (4 points)

In a few sentences, describe the difference between each of these:

```
9
'9'
"9"
'\9'
```

Coding Assignment 3.5: (4 points)

Define a function to convert lowercase letters to their uppercase counterparts. If the parameter is not a lowercase letter, it should be returned unchanged. Your function should not have 26 conditions, and should not use the built-in `toUpper` function.

Coding Assignment 3.6: (4 points)

Define a function `charToNum :: Char -> Int` which converts a digit like `'9'` into its value, `9`. Your function should do something sensible if the input is not a digit. (Don't use `read` or `digitToInt` for this.)

Coding Assignment 3.7: (2 points)

Write tests and/or propositions for the previous two functions. Test your propositions with Quickcheck if you are able to do so.

Written Assignment 3.8: (2 points)

In a sentence or two, describe what `putStr` does. What function from another programming language you know does `putStr` most resemble?

Coding Assignment 3.9: (4 points)

Define a function

onThreeLines :: String -> String -> String -> String

which takes three strings and returns a single string. When that string is printed, it shows the three input Strings on separate lines. (Your function should not print anything itself, just return a String.)

Written Assignment 3.10: (4 points)

In a few sentences, define **overloading**. Give two examples of things that are overloaded in Haskell, and explain why.

Discussion Prep 3.11: (4 Points)

Include in your writeup at least three thoughts or questions you had about the chapter 3 reading.

Extra Credit 3.Special (2 Points)

Do a bit of research into what that *Unicode Consortium* is and what it does, and describe it in about a paragraph, in your own words. Cite your sources.

Preparation for the Future

Moving forward, the expectation will be that you feel comfortable with the topics covered in this assignment. If you do not, you should ask questions on Piazza and/or continue to practice with GHC. If you want more practice, try exercises 3.20 through 3.25 in your textbook. (You do not have to submit these, but I would encourage you to do them.)