CSCE 314 [Sections 595, 596, 597] Programming Languages – Hyunyoung Lee

Exercise 1 (Total 10+2 (bonus) Points)

Due date and time: 11:59 p.m. Wednesday, January 24, 2024

The goal of this exercise is for you to get familiar with the GHCi environment and the Prelude standard library functions and to try some Prelude functions. Furthermore, you should be able to fix basic formatting errors.

This exercise contains three problems (with subproblems in each problem). You will submit your answers on Canvas Gradescope.

Problem 1 (6 Points) Start GHCi and try running the codes shown on the following slides of the lecture notes "haskell-01-basics.pdf": #15 (`map` and `reverse` separately), #20 (`sqrt` with arithmetic operators), and the ten functions in slides #22–24 (so, there are total 12 functions by counting `reverse` only once). Apply each function to at least two different arguments, that is, in your answers, you should report at least two *correct* applications (without error) of each function applied to two different arguments and their respective results. ***Explain*** (briefly) your answers ***in your own words.*** For example, here are two applications of `reverse`, their respective results, and my explanation.

For reverse:

```
Prelude> reverse [3,5,7,9]
[9,7,5,3]

Explanation: reverse reverses the given argument list, so in this application
it returns the reversed list of [3,5,7,9], which is [9,7,5,3].

Prelude> reverse [1..5]
[5,4,3,2,1]

Explanation: reverse reverses the given argument list, so in this application
it returns the reversed list of [1..5], which is [5,4,3,2,1].
```

Problem 2 (3 Points) On slide #15 of "haskell-01-basics.pdf" `map` and `reverse` are used together to achieve a more complex task. There, `reverse` is given as a function argument to `map`, which can be explained as follows. `map` accepts two arguments, the first one of which is a function and the second one a list, and then it applies the first argument (the function) to each element in the second argument (the list). Thus,

```
Prelude> map reverse ["abc","def"]
["cba","fed"]
```

Explanation: `map` applies the function reverse to each of the two strings `"abc"` and `"def"`, keeping the reversed strings in a list in the same order as the argument list, so the final result is a list that contains each string reversed, `["cba","fed"]`. The above expression can be rewritten as below without explicitly using `map` but yields exactly the same results.

```
Prelude> [reverse "abc", reverse "def"]
["cba","fed"]
```

Try each of the following applications of `map` in GHCi and report the result, explain the result, and rewrite it without explicitly using `map` and also report the result as shown above.

2.1 `Prelude> map length ["Howdy","all"]`

2.2 `Prelude> map head ["What","a","lovely","day"]`

Here, notice that a list containing characters as its elements is always shown as a string, i.e., `['a','b']` is output as `"ab"` whose type is `String` (which is a type synonym for list of `Char`s, `[Char]`). We discuss Haskell types (including the `String` type) in Chapter 3.

2.3 `Prelude> map sum [[1..10],[1,6,7],[1..100]]`

Problem 3 (3 Points) Study slides #25–34 of the lecture notes "haskell-01-basics.pdf" to find the answers to this problem. Fix the syntax errors in the program below. For this problem, you need to create your own Haskell script (e.g., exer1.hs file) that contains the program below.

```
N = a 'div' length xs
    where
        a = 10
       xs = [1,2,3,4,5]
```

There are currently three syntax errors. Report each compile error, and explain how you fixed it (the explanation should be clear and to the point).

Once you fix all three errors, change the `div` function with a division operator `/`, which will cause a compile error. Report the error, and explain why this error was caused. To do so, look up in the Prelude library how the `div` function and the division operator (`/`) are defined.

Have fun!