COMP 141: Haskell — Part 1

Instructions: In this exercise, we are going to review a bunch of Haskell structures.

(1) Set the GHCi prompt to `haskell$`. What command did you use?

(2) Calculate the floating point division of 12 to -5. What command did you use? What is the result?

(3) Calculate the integer division of 12 to -5. What command did you use? What is the result?

(4) Define function `least` that receives three numbers and returns the least value, in the following cases.

  (a) Use `min` function.

  (b) Use if-expressions.

(5) Translate the following arithmetic and boolean expressions to Haskell, and compute the results.

  (a) $3 \times 4.5 - 6.2/5.8$

  (b) $(true \wedge false) \vee (true \vee \neg false)$

(6) Define function `circle` that receives string `action` and number `radius`. If action is `"circumference"`, then the function returns circumference of the circle based on the radius. If action is `"area"`, then the function returns area of the circle based on the radius. If action is neither `"circumference"` nor `"area"`, then the function returns 0.0. For instance, `circle "area" 2.3` returns `16.619025137490002`, whereas `circle "circumference" 2.3` returns `14.451326206513047`. *Hint*: You can use built-in value `pi` in your calculations.

(7) Define function `cylinder` that receives string `action` and two numbers: `radius` and `height`. If action is `"volume"`, then the function returns volume of the cylinder based on the radius and height. If action is `"area"`, then the function returns area of the cylinder based on the radius and height. If action is neither `"volume"` nor `"area"`, then the function returns 0.0. For instance, `cylinder "volume" 1 1` returns `3.141592653589793`, whereas `cylinder "area" 1 1` returns `12.566370614359172`. *Note*: You **must** use function `circle` in both volume and area calculation of cylinder.

(8) Define function `gCD` that receives two numbers as input and returns greatest common divisor of them. For instance, `gCD 15 10` returns `5`, whereas `gCD 15 12` returns `3`. *Note*: There is a built-in library function `gcd` in Haskell. Do not use that. *Hint*: You should define the function recursively.

(9) Define function `isDivisible` that receives two numbers as input and returns `True` if the first input is divisible to the second input. Otherwise, it returns `False`. For instance, `isDivisble 6 4` returns `False`, whereas `isDivisble 6 3` returns `True`.

(10) Primary U.S. interstate highways are numbered 1-99. Auxiliary highways are numbered 100-999, and service the primary highway indicated by the rightmost two digits. Thus, I-405 services I-5, and I-290 services I-90. *Note*: 200 is not a valid auxiliary highway because 00 is not a valid primary highway number.

  Define function `highway` that receives the highway number and indicates (as a string) whether it is a primary, auxiliary, or invalid highway number. If auxiliary, indicate what primary highway it serves.

  Hint: In order to append two strings, you can use function `++`. Also, to turn a number to a string, you can use function `show`. For example, `"hello" ++ show 5` returns string `"hello5"`. How? `show 5` returns string `"5"`. Thus, `"hello" ++ "5"` returns string `"hello5"`.

  Note: Use function `isDivisble` from above in the definition of function `highway`.

  Here is a demo of the function in ghci for different inputs:

```
ghci> highway (-8)
"Not a valid interstate highway number"
ghci> highway 1632
"Not a valid interstate highway number"
ghci> highway 700
"Not a valid interstate highway number"
ghci> highway 189
"Auxiliary interstate highway, serving I-89"
ghci> highway 89
"Primary interstate highway number"
```