Basic Types, Haskell Bird's Eye View: Practice Exercises

Exercice 1

Type declaration anatomy, querying type information

1. What is the meaning of the symbol **::** in Haskell?
2. What is the keyword ***data*** used for in Haskell?
3. Define ***data constructor*** and **type constructor** in Haskell
4. What is the type of the expression `2 + 3 == 5?` What would we expect as a result?
5. What is the type and expected result value of the following:

```
Prelude> let x = 5
Prelude> x + 3 == 5
```

Exercice 2

Type definition

1. The students of an institute are divided into teams according to their registration number. There are three possible teams: RED, GREEN and BLUE. The assignment takes place with the following criterion: the student with matricula 1 goes to RED team, the one with matricula 2 in the GREEN, the one with matricula number 3 in the BLUE, the one with matricula number 4 in the RED, that one with 5 in GREEN etc. How do we write in Haskell the type signature of the type that well describes the team?

2. The appreciation corresponding to a student's score varies as follows: "Insufficient" if it is less than 18, "Just enough" (if the score is 18), "Low" (if the score is between 19-20), "Medium" (if the score is between 21-23), "Good" (if the score is between 24-26), "High" (if the score is between 27-29), "Maximum" (if the score is 30) "Impossible" (in others cases)
Define a type that faithfully captures the concept of appreciation corresponding to a student score.

Exercice 3

Converting from Integral to Num

1. Query the type of the function fromIntegral
2. The operation `6 / length [1, 2, 3]` throws an error. Explain the error.
3. How can you correct that error?

Exercice 4

Understanding basic syntax: spotting errors and interpreting them

In your REPL enter the following:

```
coursDeStageHaskell = ["Haskell", "Plutus", "Elm"]
coursSuplementaires = ["Architecture", "DLT"]
tousLesCours        = [coursDeStageHaskell, coursSuplementaires]
```

Below are some bits of code. Which will work? Why or why not? If they will work, what value would these reduce to?

```
Prelude> length coursDeStageHaskell == 2
Prelude> length [1, 'a', 3, 'b']
Prelude> length tousLesCours + length coursDeStageHaskell
Prelude> (8 == 8) && ('b' < 'a')
Prelude> (8 == 8) && 9
Prelude> (8 == 0) || (9 < 10)
```

Exercice 5

Function definition, Parameters and result

Write a program that takes as input a list of integers and returns the average of the number in the list.

Exercice 6

List type basics

you will gain more by working out the answer before you check what GHCi tells you, but be sure to use your REPL to check your answers to the following exercises. Also, you will need to have the `coursDeStageHaskell`, also, and `tousLesCours` code from above in scope for this REPL session. For convenience of reference, here are those values again:

```
coursDeStageHaskell = ["Haskell", "Plutus", "Elm"]
coursSuplementaires = ["Architecture", "DLT"]
tousLesCours        = [coursDeStageHaskell, coursSuplementaires]
```

length is a function that takes a list and returns a result that tells how many items are in the list.

1. Given the definition of length above, what would the type signature be? How many arguments, of what type does it take? What is the type of the result it evaluates to?
2. What are the results of the following expressions?
    a. length [1, 2, 3, 4, 5]
    b. length [(1, 2), (2, 3), (3, 4)]
    c. length tousLesCours
    d. length (concat tousLesCours)
3. Rewrite the ***length (concat tousLesCours)*** expression using the (**.**) operator

Exercice 7

Determining built in functions' types

Match the function names to their types.

1. Which of the following types is the type of show?
    a. show a => a -> String
    b. Show a -> a -> String
    c. Show a => a -> String

2. Which of the following types is the type of (==)?
    a. a -> a -> Bool
    b. Eq a => a -> a -> Bool
    c. Eq a -> a -> a -> Bool
    d. Eq a => A -> Bool
3. Which of the following types is the type of fst?
    a. (a, b) -> à
    b. b -> à
    c. (a, b) -> b
4. Which of the following types is the type of (+)?
    a. (+) :: Num a -> a -> a -> Bool
    b. (+) :: Num a => a -> a -> Bool
    c. (+) :: num a => a -> a -> a
    d. (+) :: Num a => a -> a -> a
    e. (+) :: a -> a -> a

Exercice 8

Re-using existing function within our function

1. The library function last selects the last element of a non-empty list; for example,

```
last [1,2,3,4,5] = 5
```

rewrite the function ***last*** in terms of ***reverse*** and ***head*** using the (**.**) composition operator

2. Write a function that tells you whether or not a given String (or list) is a palindrome. Here you'll want to use a function called reverse, a predefined function that does what it sounds like.

```
reverse :: [a] -> [a]
reverse "blah"
"halb"
```

Exercice 9

Parentesisation, precedence, conventions, scope

Parentheses allow us to set expressions priorities. In other terms, they define the order of execution.

1. Parenthesise the following numeric expressions and determine the precedence for the operator (*, +, - , / , ^)
    a. 2^3*4
    b. 2*3+4*5
    c. 2+3*4^5

2. The script (***Script.hs***) below contains three syntactic errors. Correct these errors and then check that your script works properly by loading (***:load***) it into GHCi. Explain the meaning of each error.

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

Exercice 10

Spotting out expressions' types

1. What are the types of the following values? Verify your answer in the REPL
    a. ['a' , 'b' , 'c']
    b. ('a', 'b' , 'c')
    c. [(False, 'O'), (True, '1')]

    d. ([False, True], ['0','1'])
    e. [tail, init, reverse]
2. Write down definitions that have the following types; it does not matter what the definitions actually do as long as they are type correct.
    a. bools :: [Bool]
    b. nums :: [[Int]]
    c. addTree :: Int -> Int -> Int -> Int
    d. copy :: Double -> (Double, Double)

3. What are the types of the following functions?
   Hint: take care to include the necessary class constraints in the types.
    a. secondElem xs = head (tail xs)
    b. swap (x, y) = (y, x)
    c. pair x y = (x, y)
    d. double x = x * 2

Exercice 11

Completing the syntax, understanding the let … in ... and where constructs

Given the following incomplete code:

```
multiplyByTwo n =
    let ... = 2
    in n * factor
```

1. Complete the missing piece in the following program.
2. Copy and paste into a file (***MyLetInConstruct.hs***) then load and run it in the REPL.
3. Determine and explain the type signature of the function multiplyByTwo
4. When is it appropriate to use the ***let … in …*** construct?

Exercice 12

Completing the syntax, understanding the where ... and where constructs

Given the following incomplete code:

```
multiplyByTwo n = n * factor
    where ... = 2
```

1. Complete the missing piece in the following program.
2. Copy and paste into a file (***MyWhereConstruct.hs***) then load and run it in the REPL.
3. Explain the type signature of the function multiplyByTwo
4. When is it appropriate to use the ***where …*** construct?