Haskell – Lab 2

Prepared by Dr. Wooi Ping Cheah

Solution for the Exercises from Chapter 2 – First Steps

```
-- Chapter 2

-- Slide 20
a = b + c
  where
    {b = 1; c = 2}
d = a * 2
```

```
a = b + c
  where
    b = 1
    c = 2
d = a * 2
```

```
-- Slide 21
n = a `div` length xs
  where
    a = 10
    xs = [1,2,3,4,5]


myLast1 xs = xs !! (length xs-1)
myLast2 xs = head (reverse xs)


myInit1 xs = take (length xs-1) xs
myInit2 xs = reverse (tail (reverse xs))
```

Exercises
from
Chapter 3 – Types and Classes

(1) What are the types of the following values?

```
['a','b','c']

('a','b','c')

[(False,'0'),(True,'1')]

([False,True],['0','1'])

[tail,init,reverse]
```

Hint: For the first four, please refer to Chapter 3, Pages 6-10.
The last one is a list of polymorphic functions, and each function is a mapping from a list to another list. Please refer to Pages 18-20.

(2) What are the types of the following functions?

```
second xs = head (tail xs)

swap (x,y) = (y,x)

pair x y = (x,y)

double x = x*2

palindrome xs = reverse xs == xs

twice f x = f (f x)
```

(3) Check your answers using GHCi.

Hint: For Question (2), please refer to Pages 18-23.

For Question (2), palindrome is an overloaded function because **xs** should be an Equality type.

For Question (2), determining the type of the function **twice** is quite challenging.

For Question (3), create a Haskell script that contains the above 6 function definitions, and save the file.

Switch to the ghci mode, load the script file, use the type command to find out the type of a function.

Example: When you enter **:type second**, you will get an answer **second :: [a] -> a**.

Exercises
from
Chapter 4 – Defining Functions

(1) Consider a function safetail that behaves in the same way as tail, except that safetail maps the empty list to the empty list, whereas tail gives an error in this case. Define safetail using:

(a) a conditional expression;
(b) guarded equations;
(c) pattern matching.

Hint: the library function null :: [a] → Bool can be used to test if a list is empty.

Hint: There is a function definition for **tail** on Page 10 of Chapter 4. Unfortunately, this function will fail when the argument is an empty list (i.e., **ghci> tail []** will generate an error message).
Write a function **safetail** so that it will return an empty list (i.e., **[]**) instead of an error message.

(2) Give three possible definitions for the logical or operator (||) using pattern matching.

Hints: In Chapter 4, Pages 6-7, we show you 3 ways of defining the logical **and** operator (**&&**). Now, we want you to define the logical **or** operator (**||**) using a similar approach.

Note: Since || is a standard operator in Haskell Prelude, you may use **@@** to avoid any conflict.

(3) Redefine the following version of (&&) using conditionals rather than patterns:

```
True && True = True
_    &&  _   = False
```

(4) Do the same for the following version:

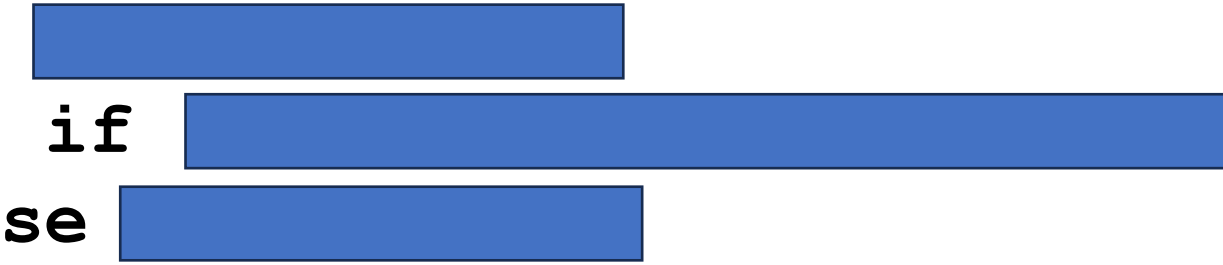
```
True  && b = b
False && _ = False
```

Hint: Questions 3 and 4 provide two versions of definition for the logical (&&) based on Pattern Matching (refer to Pages 5-8 of Chapter 4). You are asked to redefine them into Conditional Expressions described on Pages 1-2 in Chapter 4.

Note: Since && is a standard operator in Haskell Prelude, you may use $$ to avoid any conflict.

Conditional expressions for Questions 3 and 4 may look something like this

```
($$) :: Bool -> Bool -> Bool
a $$ b = if
           if
         else
```



```
($$) :: Bool -> Bool -> Bool
a $$ b = if
```

References
Functional Programming & Haskell

Curried Functions – Computerphile

https://www.youtube.com/watch?v=psmu_VAuiag

Learning Haskell Week03 – Conditional Expressions, Guarded Equations, Pattern Matching

https://www.youtube.com/watch?v=BRoPkOMPSOo&t=856s

Lambda Calculus – Computerphile

https://www.youtube.com/watch?v=eis11j_iGMs