CS\_205 Declarative Programming 10 October 2019

Monika Seisenberger

Lab Task 2: Types and Type classes

1. **What are the types for the following expressions?** Write the answer on this sheet and confirm your answer using Haskell [Hint: use the command :type.]

(’a’,’b’,’c’) :: (Char, Char, Char)

[’a’,’b’,’c’] :: [Char]

[(’1’, False), (’0’,True)] :: [(Char, Bool)]

([True,False], [’0’,’1’]) :: ([Bool], [Char])

# Types: Char, Int and Bool

Write a function that takes as input two lists of type char, and checks whether the two lists are equal.

(Challenge: modify the function so that it checks whether the two lists are equal at a given index n. You may assume that the user enteres a “sensible” index.)

equalList :: [Char] -> [Char] -> Bool

equalList x y = x == y

equalListIndex :: [Char] -> [Char] -> Int -> Bool

equalListIndex x y z = x!!z == y!!z

# What are the (most general) types for the following functions:

Hint: So make the task simpler at first: Think of integers, and write the type of the function, and then do the same using type variables a, b, .... Write and test the completed functions in Haskell.

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

swap :: (Int,Int) -> (Int,Int)

swap :: (b, a) -> (a, b)

middle (x,y,z) = y

middle :: (Int,Int,Int) -> Int

middle :: (a, b, c) -> b

palinedrome xs = xs == reverse xs

palinedrome :: [Int] -> Bool

palinedrome :: Eq a => [a] -> Bool

For the next ones you also need to check which of the constraints: Num a, Ord a, Eq a you need to add.

add x y z = x + y + z

add :: Int -> Int -> Int -> Int

add :: (Num a) => a -> a -> a -> -> a

ordered (x,y,z) = x<=y && y<=z

ordered :: (Int, Int, Int) -> Bool

ordered :: (Ord a) => (a,a,a) -> Bool

**Additional exercise)** a) A string is called a palindrome if it is the same when read

backwards. Examples are ”madam” or ”kajak”. Write a function ispalindrome:: String -> Bool

which takes a string and decides whether it is a palindrome.

[Hint: A string is nothing else but a list of characters. So, "Hello" is just syntactic sugar for [’H’,’e’,’l’,’l’,’o’]. Thus, you can use the usual list operations, e.g., "Hello" ++ " " ++ "World" gives "Hello World".]

b) Add a second function makepalindrome:: String -> String that leaves a given string

unchanged if it is a palindrome, and appends (in Haskell: ++) its reverse (to turn it into a palindrome) otherwise. Note the second function should make use of the function ispalindrome, to avoid code duplication.

Example: kajak will yield kajak; Hello will yield HelloolleH. Hint: Syntax of an if-statement:

isZero :: Int -> Bool isZero n = if n==0

then True else False

Of course this function can be written more concise without if-statement:

isZero :: Int -> Bool isZero n = n==0

2