

*04IN1023: Introduction to functional programming*

Final—SS 2013

Universität Koblenz-Landau, FB4

Prof. Dr. Ralf Lämmel, M.Sc. Andrei Varanovich

19 July 2013

Name, Vorname	
Matrikel-Nr.	
Email	.....@uni-koblenz.de
Studiengang	<input type="checkbox"/> BSc Inf <input type="checkbox"/> BSc CV <input type="checkbox"/> .....
Prüfungsversuch	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3

Hiermit bestätige ich, dass ich zur Klausur angemeldet und zugelassen bin.

Eine falsche Angabe wird als Täuschungsversuch gewertet.

Unterschrift: \_\_\_\_\_

**Korrekturabschnitt**

Aufgabe	Punkte (0-2)
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

## 1 “Simple algorithms”

Define a function *sum* to sum up a list of ints. Please use pattern matching on lists. Here is a demo:

```
> sum [1,2,3]
6
```

## 2 “Simple data models”

Declare a data type for ‘boxes’ as follows. A ‘box’ contains items and it has a certain size (i.e., a float for the side length). An ‘item’ can be either another (presumably smaller sized) ‘box’ or ‘filler material’ measured in terms of a float for its weight. (Thus, boxes may be arbitrarily nested.)

### 3 “Local scope”

Consider the following code:

```
test x y z = smaller x y && smaller x z
smaller x y = x < y
```

Transform the code such that local scope is used for the definition of *smaller*, i.e., *smaller* becomes a local function of *test*. The local definition should only have a single argument. Hint: note that *smaller* is invoked both times with the same first argument.

## 4 “Parametric polymorphism”

Define a polymorphic function *shiftLeft* including its function signature for shifting all elements of a list to the left with the original head becoming the last element of the resulting list. Here is an illustration:

```
> shiftLeft []
[]
> shiftLeft [1]
[1]
> shiftLeft [1,2]
[2,1]
> shiftLeft [1,2,3]
[2,3,1]
```

## 5 “Higher-order functions”

Define the polymorphic function *maybe* which dispatches on a *Maybe* value as demonstrated here:

```
> maybe 0 (1+) Nothing
0
> maybe 0 (1+) (Just 41)
42
```

That is, ‘*maybe b f v*’ evaluates to *b* if *v* is *Nothing* and it evaluates to ‘*f a*’ if *v* is ‘*Just a*’. (This is the standard *maybe* function.)

## 6 “Monoids”

Consider the following code:

```
instance Monoid [a] where
  mempty = []
  mappend = ++
```

What does the shown monoid instance describe? *Please, be concise: 140 characters or less.*

## 7 “Functors”

Consider the following data-type declaration for some sort of trees with one constructor for empty trees and another constructor for forking trees with an associated list of values of the parameter type of the datatype constructor:

```
data Tree a = Empty | Fork [a] (Tree a) (Tree a)
```

Describe an instance of the type class *Functor* with its member function *fmap*, as needed for the trees at hand.



## 8 “Reasoning”

Here is an attempt at formulating a property for testing *drop*. (Remember, *drop* is the function which drops (‘removes’) the given number of elements of a list.)

```
import Test.QuickCheck
```

```
prop_drop x l = length (drop x l) > length (drop (x+1) l)
```

This ‘property’ is not universally true. Give an application of the ‘property’ for which it returns *False*.

## 9 “Lazy evaluation”

Consider the following definition of the factorial function:

```
factorial x = product [1..x]
```

Now, also consider the following definition of all non-zero natural numbers:

```
nats = nats' 1
  where
    nats' x = x : nats' (x+1)
```

(Clearly, *nats* denotes an infinite list.) Re-define the factorial function to use *nats* rather than the “..” notation. Hint: you may also need the *Prelude* function *take* for taking (‘selecting’) a given number of elements of a list.

## 10 “Monads”

Consider the following definition of *return* of a *State* monad.

```
-- Data type for the state monad
newtype State s a = State { runState :: s -> (a,s) }

-- Monad instance for State
instance Monad (State s)
  where
    return x = State (\s -> (x, s))
    c >>= f = ... -- omitted for brevity
```

What does the definition of *return* model? *Please, be concise: 140 characters or less.*