

Prof. Dr. Jürgen Giesl
Peter Schneider-Kamp
Stephan Swiderski

Exercises *Functional Programming* – Sheet 1

Solutions will be collected until Wednesday, Apr 18, 2007 in the exercise course.

Exercises can be solved both in English and in German.

The HASKELL interpreter HUGS can be download at www.haskell.org/hugs.
You can use „:l filename“ to load a HASKELL file, „:r“ to reload a changed file
and „:q“ to quit the interpreter. Your programs can be tested by typing arbitrary
HASKELL-expressions at the prompt.

Exercise 1 (1+1 points)

- (a) Give examples of functions with the following types:
- (i) $(\text{Int} \rightarrow \text{Bool}) \rightarrow \text{Int}$
 - (ii) $(\text{Int} \rightarrow \text{Int}) \rightarrow \text{Int} \rightarrow \text{Int}$
 - (iii) $[\text{Int}] \rightarrow [\text{Int}] \rightarrow \text{Int}$
- (b) Suppose that f and g have the type $(\text{Bool} \rightarrow \text{Bool}) \rightarrow \text{Bool} \rightarrow \text{Bool}$.
Let h be defined by $h\ x\ y = f\ (g\ x)\ (g\ x\ y)$. What is the type of h ?

Exercise 2 (2 points)

Which of the following equations between two lists are correct? Explain your answers.

- (a) $x : (xs : ys) = x : xs : ys$ (b) $x : ([x] ++ ys) = x : x : ys$
(c) $x : [] : xs = [x] : xs$ (d) $(xs ++ x) : ys = xs ++ [x] ++ ys$

The operation `++` concatenates two lists. For example: `[1,2,3] ++ [2,3] = [1,2,3,2,3]`.

Exercise 3 (2+1+2 points)

- (a) Write a HASKELL function that sorts a list of integers:

```
sort :: [Int] -> [Int]
```

For example `sort [1,3,2,2] = [1,2,3]`. Duplicates should be eliminated.

(b) Write a HASKELL function to compute the median of a list:

```
median :: [Int] -> Int
```

For example `median [4,6,5,7,7] = 5`, because the second element of the sorted list without duplicates `[4,5,6,7]` is 5, the length n of the list `[4,5,6,7]` is 4, and we use the $\lfloor n/2 \rfloor$ -th element of the list as median. Note that the median of the empty list is not defined. To compute the median, use your `sort` function and the predefined functions `div`, `!!`, and `length`, where `div n m` computes $\lfloor n/m \rfloor$ and `[x0, ..., xn] !! m = xm`:

```
div :: Int -> Int -> Int
```

```
(!!) :: [Int] -> Int -> Int
```

```
length :: [Int] -> Int
```

(c) – Write a version of `median` with a local declaration (using `where`) that avoids the repeated evaluation of identical subexpressions:

```
medianld :: [Int] -> Int
```

– How many evaluation steps does the lazy evaluation of `medianld [1]` take? Explain the difference with regard to the lazy evaluation of `median [1]`.

Exercise 4 (3 points)

Define the HASKELL functions `*|` and `|-|` in infix notation such that the following holds:

- `[x1, x2, ..., xn] |-| [y1, y2, ..., yn]` evaluates to `[x1-y1, x2-y2, ..., xn-yn]` for all $x_i, y_i \in \mathbb{N}$,
- `x *| [y1, y2, ..., yn]` evaluates to `[x*y1, x*y2, ..., x*yn]`
- `x |-| y |-| z` is the same as `(x |-| y) |-| z`, and
- `x *| y |-| z` is the same as `(x *| y) |-| z`.

You may not use any predefined functions except for `+`, `*`, and `-`. Set the priority of `*|` and `|-|` accordingly.