Technical University of Munich
Chair for Logic and Verification
Prof. Tobias Nipkow, Ph.D.
J. Rädle, L. Stevens, K. Kappelmann

WS 2019/20
18.10.2019
Deadline: 29.10.2019, 00:00

Functional Programming and Verification

Sheet 1

IMPORTANT: You may only attend the tutorial with a running version of GHC as specified on the course website http://www21.in.tum.de/teaching/fpv/WS1920/ghc.html.

Tutorial Exercises

Exercise T1.1 Hello Haskell

1. Define a function

   ```
   threeAscending :: Integer -> Integer -> Integer -> Bool
   ```

   that returns `True` if and only if the sequence of parameters is strictly monotonically increasing.

2. Define a function

   ```
   fourEqual :: Integer -> Integer -> Integer -> Integer -> Bool
   ```

   that returns `True` if and only if all parameters are equal.

3. Evaluate the following expressions by hand, line by line:

   ```
   threeAscending (2+3) 5 (11 `div` 2)
   fourEqual (2+3) 5 (11 `div` 2) (21 `mod` 11)
   ```

Exercise T1.2 For Recursion See Recursion

1. Define a recursive function `fac :: Integer -> Integer` such that `fac n` $= n!$.
2. Define a function `sumEleven :: Integer -> Integer` such that `sumEleven n` $= \sum_{i=n}^{n+10} i$.
   Hint: use an auxiliary function.

Exercise T1.3 Maximum Fun

1. Define a recursive function

   ```
   argMax :: (Integer -> Integer) -> Integer -> Integer
   ```

   such that `argMax g n` maximises `g` in the domain $\{0, \ldots, n\}$.

2. Let `g` be the following function

   ```
   g :: Integer -> Integer
   g n = if n < 10 then n*n else n.
   ```

Examine `g` to determine when `argMax g n` $\neq$ `n`. Use your observations to write a function `argMaxG :: Integer -> Integer` that does not use `g` and satisfies the property `argMax g n = argMaxG n`. Write a QuickCheck test to check the equivalence.

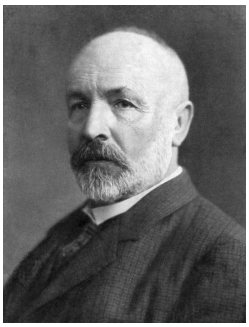
Homework

Important: You have to submit your homework on https://vmnipkow3.in.tum.de/web/. Your homework will automatically be tested and graded. You can submit as many times as you want. Your *last* submission will constitute your final grade. Read the FAQ at http://www21.in.tum.de/teaching/fpv/WS1920/faq-tests.html.

In this homework, we shall meet some of the best mathematicians in history. You need to collect 4 out of 7 points (P) to pass this sheet and become part of the mathematical society.

Exercise H1.1 Cantor's Creativity [1: 1P, 2: 1P, 3+4: 1P]

As a matter of course, Haskell knows about pairs; however, we sadly haven't learnt about them in class so far. But fear not! As shown by the great Georg Cantor, we can just encode pairs in a clever way. We do not directly follow the great Cantor's approach though but define a different encoding function proposed by the MC Sr. The following functions only need to work for natural numbers $\mathbb{N} = \{0, 1, 2, \dots\}$.



Georg Cantor

1. Define the encoding function

   ```
   myPair :: Integer -> Integer -> Integer
   ```

   such that

   `myPair x y` $= 2^x(2y + 1)$.

2. Define the inverse function

   ```
   myFst :: Integer -> Integer
   ```

   such that

   `myFst (myPair x y)` $= x$.

   Hint: Divide by 2 until the number is not divisible by 2 anymore.

3. Define the inverse function

   ```
   mySnd :: Integer -> Integer
   ```

   such that

   `mySnd (myPair x y)` $= y$.

Carl Friedrich Gauss

The only existing portrait/caricature of Adrien-Marie Legendre

Leonhard Euler

4. Write a QuickCheck test with parameters $x, y \in \mathbb{N}$ that checks whether `myFst` and `mySnd` are indeed inverse functions, that is they satisfy the equations stated above.

   Hint: You can restrict your test's domain by using the "`==>`" operator, e.g. `x >= 0 ==> x^3 >= 0`.

Exercise H1.2 Legendre the Legend [1: 1P, 2: 1P, 3: 1P, 4+5: 1P]

1. Define a function

   ```
   equivMod :: Integer -> Integer -> Integer -> Bool
   ```

   such that `equivMod n a b` returns `True` if and only if $a \equiv b \pmod{n}$; that is, a is equivalent to b modulo n. You can assume that $n \in \mathbb{N}_+ := \{1, 2, 3, \dots\}$

2. (**Competition**) Given $n \in \mathbb{N}_+$ and $q \in \mathbb{Z}$, we say that q is a *quadratic residue* modulo n if there exists $x \in \mathbb{Z}$ such that $x^2 \equiv q \pmod{n}$. Quadratic residues were first systematically treated by Carl Friedrich Gauss. Define a function

   ```
   quadRes :: Integer -> Integer -> Bool
   ```

   such that `quadRes n a` returns `True` if and only if a is a quadratic residue modulo n.

   Hint: Since $(n+m)^2 \mod n = n^2 + 2nm + m^2 \mod n = m^2 \mod n$, one only needs to search for candidates in $\{0, \dots, n-1\}$.

   This exercise was posed by the Master of Competition Junior (MC Jr.). It will be marked as part of your homework but also counts towards the competition.[1] The solution with the smallest number of tokens wins the competition. Tokens include identifiers, operators, brackets, numbers, etc.[2] The length of an identifier is irrelevant. You can download the official token counter from the lecture website.[3]

[1] http://www21.in.tum.de/teaching/fpv/WS1920/wettbewerb.html
[2] http://www21.in.tum.de/teaching/fpv/WS1920/wettbewerb.html#token
[3] http://www21.in.tum.de/teaching/fpv/WS1920/Tokenize.hs

You are allowed to use anything from the libraries specified on the website[4] as well as self-written auxiliary functions. The complete solution (including self-written functions like `equivMod` but excluding functions from external libraries) must be submitted inside the comments {-WETT-} and {-TTEW-}, for example

```
{-WETT-}
quadRes :: Integer -> Integer -> Bool
quadRes n a  = ...
{-TTEW-}
```

The Master of Competition Sr. (MC Sr.) needed 21 tokens for his solution. Show him what you got!

> **Important:** If you submit a competition exercise, you agree that we are allowed to publish your name as part of the competition on our website. If you just want to submit a competition exercise as part of your homework without taking part in the competition, you can just remove the {-WETT-}...{-TTEW-} comments of your submission.

3. Next, we implement the Legendre symbol $\left(\frac{a}{n}\right)$ as introduced by the French mathematician Adrien-Marie Legendre. Define a function

   ```
   legendre :: Integer -> Integer -> Integer
   ```

   such that

   $$\texttt{legendre n a} = \begin{cases} 1, & \text{if } a \text{ is a quadratic residue modulo } n \text{ and } a \not\equiv 0 \pmod{n} \\ -1, & \text{if } a \text{ is a not a quadratic residue modulo } n \\ 0, & \text{if } a \equiv 0 \pmod{n} \end{cases}.$$

4. Define a function `prime :: Integer -> Bool` such that, for all $n \in \mathbb{N}_+$, `prime n` returns `True` if and only if n is prime. You do not have to be clever and can just implement a naive algorithm. But of course, you are allowed to be clever to impress your tutor.
5. Given an odd prime p and $a \in \mathbb{Z}$, Euler's criterion states that

   $$\texttt{legendre p a} \equiv a^{\frac{p-1}{2}} \pmod{p}.$$

   Write a QuickCheck test with parameters p, a that checks whether your implementation of `legendre` indeed satisfies the criterion.

> *Now I feel as if I should succeed in doing something in mathematics, although I cannot see why it is so very important.*
>
> — Helen Keller

[4] http://www21.in.tum.de/teaching/fpv/WS1920/wettbewerb.html#libraries