Technische Universität München **SS 2017**
Institut für Informatik **28. 4. 2017**
Prof. Tobias Nipkow, Ph.D.
Dr. Peter Lammich

Functional Data Structures

Exercise Sheet 1

Before beginning to solve the exercises, open a new theory file named `ex01.thy` and write the the following three lines at the top of this file.

theory *ex01*
imports *Main*
begin

Exercise 1.1 Calculating with natural numbers

Use the **value** command to turn Isabelle into a fancy calculator and evaluate the following natural number expressions:

"$2 + (2{::}nat)$" "$(2{::}nat) * (5 + 3)$" "$(3{::}nat) * 4 - 2 * (7 + 1)$"

Can you explain the last result?

Exercise 1.2 Natural number laws

Formulate and prove the well-known laws of commutativity and associativity for addition of natural numbers.

Exercise 1.3 Counting elements of a list

Define a function which counts the number of occurrences of a particular element in a list.

fun *count* :: "$'a\ list \Rightarrow 'a \Rightarrow nat$"

Test your definition of *count* on some examples and prove that the results are indeed correct.

Prove the following inequality (and additional lemmas, if necessary) about the relation between *count* and *length*, the function returning the length of a list.

theorem "$count\ xs\ x \leq length\ xs$"

Exercise 1.4 Adding elements to the end of a list

Recall the definition of lists from the lecture. Define a function *snoc* that appends an element at the right end of a list. Do not use the existing append operator @ for lists.

fun *snoc* :: "'*a list* ⇒ '*a* ⇒ '*a list*"

Convince yourself on some test cases that your definition of *snoc* behaves as expected, for example run:

value "*snoc* [] *c*"

Also prove that your test cases are indeed correct, for instance show:

lemma "*snoc* [] *c* = [*c*]"

Next define a function *reverse* that reverses the order of elements in a list. (Do not use the existing function *rev* from the library.) Hint: Define the reverse of *x* # *xs* using the *snoc* function.

fun *reverse* :: "'*a list* ⇒ '*a list*"

Demonstrate that your definition is correct by running some test cases, and proving that those test cases are correct. For example:

value "*reverse* [*a*, *b*, *c*]"
lemma "*reverse* [*a*, *b*, *c*] = [*c*, *b*, *a*]"

Prove the following theorem. Hint: You need to find an additional lemma relating *reverse* and *snoc* to prove it.

theorem "*reverse* (*reverse xs*) = *xs*"

Homework 1.1 Maximum Value in List

Submission until Friday, May 5, 11:59am.

Submit your solution via https://vmnipkow3.in.tum.de. Submit a theory file that runs in Isabelle-2016-1 **without errors**.

General hints:

- If you cannot prove a lemma, that you need for a subsequent proof, assume this lemma by using sorry.
- Define the functions as simple as possible. In particular, do not try to make them tail recursive by introducing extra accumulator parameters — this will complicate the proofs!
- All proofs should be straightforward, and take only a few lines.

Define a function that returns the maximal element of a list of natural numbers. The result for the empty list shall be 0.

fun *lmax* :: "*nat list* ⇒ *nat*"

Define a function that checks whether an element is contained in a list

fun *lcont* :: "$'a \Rightarrow 'a$ *list* $\Rightarrow$ *bool*"

Show that the maximum is greater or equal to every element of the list.

lemma *max_greater*: "*lcont x xs* $\Longrightarrow$ $x \leq$ *lmax xs*"

Hint: If you see an *if then else* term in your premises, try to pass the option *split*: *if_splits* to *auto* or *simp*, e.g. *apply* (*auto split*: *if_splits*)

Prove that reversing the list does not affect its maximum. Note that we use the *reverse* function from exercise 4 here.

lemma "*lmax* (*reverse xs*) = *lmax xs*"

Hint: Induction. You may need an auxiliary lemma about *lmax* and *snoc*.