Exercise 1 Functions and their properties in Isabelle/HOL

Please do not use the append operator ‘op @’ or any other predefined functions on lists for this exercise.

a) (Prepare!) Write a function appendRight :: 'a list ⇒ 'a ⇒ 'a list, which appends a single element at the end of a list.

b) (Prepare!) Write a function reverse :: 'a list ⇒ 'a list, which reverts a list.

c) (Prepare!) Write a function replace :: 'a ⇒ 'a ⇒ 'a list ⇒ 'a list, where the call replace x y l should return a list in which all occurrences of x in l are replaced with y.

d) (Prepare!) Write a function forall :: ('a ⇒ bool) ⇒ 'a list ⇒ bool, which calculates whether all elements of a list satisfy the given predicate.

e) (Prepare!) Write a function exists :: ('a ⇒ bool) ⇒ 'a list ⇒ bool, which calculates whether some element of a list satisfies the given predicate.

f) Prove or disprove the following properties of the functions appendRight, reverse and replace:
   • replace x y (replace x y l) = replace x y l
   • y ≠ x −→ replace x z (replace x y l) = replace x y l
   • replace y z (replace x y l) = replace x z l
   • reverse (replace x y l) = replace x y (reverse l)

g) Prove the following properties of the functions forall and exists:
   • forall (λx. P x ∧ Q x) l = (forall P l ∧ forall Q l)
   • exists (λx. P x ∨ Q x) l = (exists P l ∨ exists Q l)
   • exists P (map f l) = exists (P o f) l
   • forall P (reverse l) = forall P l
   • exists P (reverse l) = exists P l

For additional exercises in functional programming (e.g. in ML), please refer to the slides, exercise sheets and solutions of the lecture “Software-Entwicklung I” from the winter term 2008/09. (In the winter 2009/10 and later we did Haskell in the beginners course.)

You can also find a lot of suggestions for functions on lists in the documentation of the Haskell Data.List module [http://haskell.org/ghc/docs/latest/html/libraries/base-4.5.0.0/Data-List.html](http://haskell.org/ghc/docs/latest/html/libraries/base-4.5.0.0/Data-List.html)

Please do not hesitate to ask us, if you encounter any problems when implementing such functions.
Exercise 2  Datatypes and Properties in Isabelle/HOL

On sheet 1 we defined a datatype for binary trees. Define some additional function and prove some of their properties.

a) (Prepare!) Define the functions root, leftmost and rightmost on trees, which return the respective values for non-empty trees and are undefined otherwise.

b) (Prepare!) Define a function mirror that returns the mirror image of a binary tree.

c) Prove or disprove the following theorems:
   • $t \neq \text{Empty} \rightarrow \text{last (inOrder } t) = \text{rightmost } t$
   • $t \neq \text{Empty} \rightarrow \text{hd (inOrder } t) = \text{leftmost } t$
   • $t \neq \text{Empty} \rightarrow \text{hd (preOrder } t) = \text{last (postOrder } t)$
   • $t \neq \text{Empty} \rightarrow \text{hd (preOrder } t) = \text{root } t$
   • $t \neq \text{Empty} \rightarrow \text{hd (inOrder } t) = \text{root } t$
   • $t \neq \text{Empty} \rightarrow \text{last (postOrder } t) = \text{root } t$

d) Suppose that $x\text{Order}$ and $y\text{Order}$ are tree traversal functions chosen from preOrder, postOrder, and inOrder. Examine for which traversal functions the following formula holds:
   
   $x\text{Order (mirror } xt) = \text{rev (yOrder } xt)$

Exercise 3  Binary Search Tree and its Properties

On the last sheet we defined a binary tree and a search function $\text{findT}$. Now we consider a subset of these trees: binary search trees containing natural numbers. A tree is a search tree if for every node all numbers found in the left subtree are smaller or equal to the node’s value and all numbers in the right subtree are larger.

a) Write a predicate $\text{is_search_tree} :: \text{nat tree } \Rightarrow \text{bool}$ that checks if a given binary tree is a search tree.

b) Write a function $\text{insertST} :: \text{nat tree } \Rightarrow \text{nat } \Rightarrow \text{nat tree}$ that inserts an item into a binary search tree.

c) Write a function $\text{findST} :: \text{nat tree } \Rightarrow \text{nat } \Rightarrow \text{bool}$ that searches for an item in the search tree “efficiently”.

d) Prove $\text{insertST}$ preserves the search tree property.

e) Prove that for a search tree $\text{findST}$ and $\text{findT}$ return the same result.
Exercise 4 Sets as Functions in Isabelle/HOL

Consider the following type synonym:

```markdown
type_synonym 'a myset = "a ⇒ bool"
```

The idea behind this definition is that sets can be represented by their characteristic function, i.e., the function which decides for each element if it is in the set or not.

Define the following constants for our new type:

1. The **empty** set.
2. The **insert** and **delete** function on sets.
3. The **union** and **intersection** on sets.
4. The set of all even integers.

Exercise 5 Using and Extending the Simple Theorem Prover

You can find the SimpleTheoremProver2.thy used in the lecture on our website. We want to use this simple prover to prove the theorem from Exercise 1a on Sheet 1.

a) Download the file and run it through Isabelle/HOL. Think about the existing proof in the file and how it is done.

b) Extend the data type for formulas to support the logical operators **And** and **Or**. Remember to also adjust the functions which are using the data type!

c) The theory so far has only four rules. Add further rule definitions from the natural deduction calculus, such that you can do the proof of the sequent from Exercise 1a.

d) Proof the sequent: ⊢ (a ∨ (b ∧ c)) → ((a ∨ b) ∧ (a ∨ c))

Exercise 6 Nondeterministic Finite State Machines (Hand in!)


a) Define a type which represents nondeterministic finite state machines.

b) Define a function that checks if a value of the type above is a valid finite state machine and contains at least one final state.

c) Define a function that checks if a given finite state machine accepts a word.

d) Define a finite state machine that accepts the language described by the regular expression (a|b)*ab*.

e) Prove that your machine accepts the word aab.