Prof. Dr. A. Poetzsch-Heffter
M.Sc. Peter Zeller
Dipl.-Inf. C. Feller

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Exercise Sheet 11: Specification and Verification with Higher-Order Logic (Summer Term 2014)

Exercise 1 Simpl

In this exercise we will use the Simpl program verification framework.

Download the archive `Sheet11_theories.zip` unzip it into an arbitrary directory and load the file `Sheet11_simpl.thy`.

Some hints for working with Simpl:

- The types of variables are defined by two records. One for local variables and one for the heap. Every variable name in the record has to end with `_'` but when referencing this variable you don't write down this suffix but prepend ´. The record that contains the global variables must start with `globals_`. The record that contains the local variables has to extend the record `'g state`.
- You can define procedures by using the command `procedures`. Procedure definitions have the form `procedures name(p,q|r) = "..."` where `p` and `q` are parameters and `r` is the return value.
- The language has the following constructs with standard semantics:
  - ´`x :== e`
  - `c1;;c2`
  - `IF b THEN c1 ELSE c2 FI`
  - `r :== CALL name(e1,e2)`
  - `WHILE b DO c OD`
  - ´`x`→´`next` (dereference x and access field next)
- Expressions in the language are arbitrary Isabelle expressions.
- ´`x` refers to the value of variable `x` in the current state.
- ⇗σ⇖`x` refers to the value of variable `x` in state σ.
- {|σ`. P`|}is an assertion that P holds and the current state is σ.
- To use the verification condition generator of Simpl use the command **`apply`** `vcg`. If you want to prove something about recursive procedures use **`apply`**`(hoare_rule HoarePartial.ProcRec1).`

a) Prove the examples about swap, sum and fib given in the theory file using Simpl.

b) Write a Simpl procedure which recursively calculates the minimum number in a nonempty, singly linked list. Then prove the correctness of the procedure.

c) Write a Simpl procedure that takes a singly linked list of integers and adds 1 to every element of the list using recursion. Give an appropriate specification for this example. Then show that this specification holds for your program. *Hint: You can extend the heap abstraction* `List` *in order to simplify the proof.*

d) Create new records represent binary trees in Simpl. Similar to the singly linked lists the trees should be able to store integers. Define a heap abstraction for binary trees and show that the abstraction works for a simple example tree of size 4.

e) Write a Simpl procedure that takes a tree (as defined above) and adds 1 to every element in the tree. Give an appropriate specification for this example. Then show that this specification holds for your program.