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 4: Specification and Verification with Higher-Order Logic (Summer Term 2014)

Please prepare the marked tasks for the exercise on Wednesday, May 28, 2014
Submit your solutions to the hand-in tasks before Monday, June 9, 2014

Exercise 1 Quicksort

In the lecture we have seen a definition of a quicksort function `qsort` in Isabelle and proven its termination. In this exercise we want to prove its correctness. Download the file `Sheet4_qsort.thy` to obtain the theory used in the lecture.

a) (Prepare!) Explain informally why the `qsort` function is a correct sorting function.

b) Define a property `isPermutation :: 'a list ⇒ 'a list ⇒ bool` that indicates if two lists are permutations of each other. For this it might be helpful to define a function that counts the elements in a list (`count :: 'a list ⇒ 'a ⇒ nat`).

c) Show that the result of the `qsort` function is a permution of its input.

d) Define a property `sorted :: ('a::linorder) list ⇒ bool` that indicates if a list is sorted in ascending order.

e) Show that the result of the `qsort` function is sorted.

Exercise 2 Verifying Haskell (Hand in!)

Download the file `Sheet4_exprsimp.hs` from the website. This Haskell program parses expressions given by the user and simplifies them. Expressions consist of numbers, variables, parenthesis and additions. You can run the program with the following command:

```
runhaskell Sheet4_exprsimp.hs
Enter expression to simplify (q to quit):
1+2+-4+(3+-3+x)+y+z+3
expr = (((((1 + 2) + -4) + ((3 + -3) + x)) + y) + z) + 3)
simp = ((((-1 + x) + y) + z) + 3)
```

In this exercise you should verify that the simplify function works correctly. You do not have to understand the other parts of the Haskell code. If you find any bugs during your verification work, please fix them.

a) Rewrite the datatype definition `Exp` and the function `simplify` in Isabelle. Keep the Isabelle code as close to the original code as possible.

b) Define the semantics of an expression by writing a function `sem :: "(string ⇒ int) ⇒ exp ⇒ int"`, which calculates the value of an expression for a given variable assignment.

c) Show that using the `simplify` function does not change the semantics of an expression.

d) Formulate the following property: `simplify` always returns an expression which is as small as possible.

e) Disprove the above property by showing that there is an expression for which the result of `simplify` is not optimal.

f) Generate Haskell code from your Isabelle theory using the following steps:

1. Create a folder named “hs” next to your theory file.
2. Adjust the header of your theory file as follows:

```
theory Sheet4_Haskell
imports
Main
"~~/src/HOL/Library/Code_Target_Int"
"~~/src/HOL/Library/Code_Char"
begin
```

3. Use the following line in your theory file to export the code:

```
export_code simplify in Haskell  module_name Simplify file "hs/"
```

g) (optional) Adjust the original Haskell code, so that the generated code is used for the simplification instead.

h) (optional) Define a function `simplify2` which computes a minimal simplified expression.