ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

D. Basin and P. Müller

Formal Methods and Functional Programming

Exercise Sheet 8: Motivation and Induction

Submission deadline: April 23rd, 2012

Note that the tutors for exercise groups will mostly be different for the second half of the course. Please have a look at the course web page (`http://www.infsec.ethz.ch/education/ss2012/fmfp`) to see who is your tutor. However, the times and rooms of exercise sessions remain the same. For questions about the new exercise groups, please contact Malte Schwerhoff (`malte.schwerhoff@inf.ethz.ch`).

Please submit your solution before **9:15am** on the submission date specified above. Solutions can be submitted via e-mail or by using the boxes in front of **CAB F 51.1**. Make sure that the first page (and preferably each sheet) always contains your name, the exercise sheet number as well as your tutor's name and the weekday (Tuesday or Wednesday) of your exercise group. Don't forget to staple your pages if you submit more than one page.

Assignment 1

Write a program - in your preferred imperative programming language - that computes $\lceil \sqrt[x]{y} \rceil$, for integers $x > 0$ and $y \geq 0$. Give textual arguments justifying the correctness of your program.

The notation $\lceil x \rceil$ denotes the "ceiling" function applied to x, i.e., the number x rounded up to the nearest integer. More precisely:

$$\forall x \in \mathbb{R} \cdot \ \lceil x \rceil \in \mathbb{Z} \ \wedge \ \lceil x \rceil - 1 < x \leq \lceil x \rceil$$

You should restrict yourself to using only the basic arithmetic operations (addition, substraction, multiplication, division) on integers (i.e., no built-in mathematical procedures!).

Hint: You might want to first look at the simplified problem where $x = 2$ and then generalize your solution to $x > 0$.

Hint: You might also want to have a look at the slides of "Einführung in die Programmierung" by Bertrand Meyer that are concerned with loop invariants and loop variants.

Assignment 2

Consider the following definition (*Peano numerals*) of the natural numbers and the addition operation defined on them:

```
data Nat  =  Zero
          |  Succ Nat

plus Zero n      = n
plus (Succ m) n  = Succ (plus m n)
```

Prove the associativity of `plus`, i.e.,

$\forall x, y, z \in$ `Nat`$\cdot$ `plus` x `(plus` y z`) = plus (plus` x y`)` z.

Assignment 3

Using the same definitions as the previous question, consider the following less-than-or-equal-to predicate defined as:

```
leq Zero n             = true
leq (Succ m) Zero      = false
leq (Succ m) (Succ n)  = leq m n
```

Prove the following property of the `leq` predicate:
$\forall m, n \in$ `Nat`$\cdot$ `(leq (plus` m n`)` m `= true)` $\Rightarrow$ n `= Zero`

Assignment 4

The following is a Haskell function that computes the sum of the items in a list, using tail recursion:

```
sum l = sumaux l 0
  where
    sumaux[] i = i
    sumaux(x:xs) i = sumaux xs (x+i)
```

Prove the correctness of the function, i.e., show $\forall l \in$ `[Int]`$\cdot P(l)$ where

$$P(l) \equiv \texttt{sum}\ l = \sum_{i=0}^{(\texttt{length}\ l)-1} l\,!!\,i$$

The notation `!!` denotes the element i of list l (starting from index 0).

Hint: Is $P(l)$ an adequate induction hypothesis for your proof?

Assignment 5 - Headache of the week

Consider the Haskell quick sort program:

```
qsort[] = []

qsort(p:xs) = qsort[x | x<-xs, x<=p] ++ [p] ++ qsort[x | x<-xs, x>p]
```

Prove formally that `qsort l` is sorted, assuming `l` is of finite length. You should first state formally what it means for a list of integers to be sorted.

You may use without proof the following lemmas:

Lemma 1: `length [x|x <- l, P(x)]` $\leq$ `length l`
(the list resulting from a list comprehension is no longer than the original list).

Lemma 2: `l1 = [x|x <- l, P(x)]` $\Rightarrow$ $\forall i \in \mathbb{N} : 0 \leq i <$ `length(l1)` $\Rightarrow$ `P(l1!!`i`)`
(every element of the result of a list comprehension, satisfies the property used to filter the original list).

You may also use as lemmas any properties of `length`, list append (++), list indexing (!!) and list comprehension that you need, provided that you state them clearly first, and provided that they are true!