Last Name: ______________________

First Name: ______________________

Matriculation Number: ______________________

| Exercise | Points | Score |
|---|---|---|
| Single Choice | 6 | |
| Well-Definedness of Functional Programs | 31 | |
| Verification of Functional Programs | 36 | |
| Verification of Imperative Programs | 27 | |
| $\sum$ | 100 | |

- The time limit for the exam is 100 minutes, so 1 point = 1 minute.
- The available points per exercise are written in the margin.
- Write on the printed exam for Exercises 1 and 4 and use blank sheets for the rest.
- Your answers can be written in English or German.

**Exercise 1: Single Choice** 6

For each statement indicate whether it is true (✓) or false (✗). Giving the correct answer is worth 3 points, giving no answer counts 1 point, and giving the wrong answer counts 0 points (for that statement).

1. ____ The property that a functional program $P$ is well-defined is a necessary criterion to ensure that the semantics of $P$ is well-defined.
2. ____ Whenever termination of a functional program can be proven solely by the subterm criterion, then termination can also be proven solely by the size-change principle.

**Exercise 2: Well-Definedness of Functional Programs** 31

Consider the following functional program where shuffle converts binary trees into lists and shuffles the order of the elements.

$$\begin{aligned} \mathsf{append}(\mathsf{Nil}, xs) &= xs && (1)\\ \mathsf{append}(\mathsf{Cons}(x, xs), ys) &= \mathsf{Cons}(x, \mathsf{append}(xs, ys)) && (2)\\ \mathsf{mirror}(\mathsf{Leaf}) &= \mathsf{Leaf} && (3)\\ \mathsf{mirror}(\mathsf{Node}(\ell, x, r)) &= \mathsf{Node}(\mathsf{mirror}(r), x, \mathsf{mirror}(\ell)) && (4)\\ \mathsf{shuffle}(\mathsf{Node}(\ell, x, r)) &= \mathsf{append}(\mathsf{shuffle}(\mathsf{mirror}(r)), \mathsf{Cons}(x, \mathsf{shuffle}(\mathsf{mirror}(\ell)))) && (5) \end{aligned}$$

(a) Turn the program into a well-defined functional program (without considering termination). (10)

- Add all missing data type definitions via data.
  Note: there is no unique solution.
- Provide a suitable type for the functions mirror and shuffle, assuming a suitable type for append.
- If the program is not pattern-disjoint or not pattern-complete, then modify the equations and/or add new equations to obtain a pattern-disjoint and pattern-complete program.

(b) Compute all dependency pairs of mirror and shuffle. Indicate which of these pairs can be removed by the subterm-criterion. (7)

(c) Compute the set of usable equations w.r.t. the dependency pairs of $\mathsf{shuffle}^\sharp$. It suffices to mention the indices of the equations. (4)

(d) Prove termination of shuffle by completing the following polynomial interpretation $p$. (10)

$$\begin{aligned} p_{\mathsf{shuffle}^\sharp}(t) &= \dots\\ p_{\mathsf{mirror}}(t) &= \dots\\ p_{\mathsf{Node}}(\ell, x, r) &= \dots\\ \dots &= \dots \end{aligned}$$

Hints:

- You only need numbers 0 and 1 in the polynomial interpretation.
- Use intuition and don't try to compute the constraints symbolically.
- It makes sense to start filling in suitable interpretations by looking at the constraints of the dependency pairs for $\mathsf{shuffle}^\sharp$ first, and then look at the constraints of the usable equations from the previous part.

**Exercise 3: Verification of Functional Programs** 36

Consider the following functional program on natural numbers and lists of natural numbers, where the well-known data-type definitions for $\mathsf{Nat}$ and $\mathsf{List}$ have been omitted. Observe that the definition of $\mathsf{plus}$ is not the standard one.

$$\begin{aligned}
\mathsf{plus}(x, \mathsf{Zero}) &= x \\
\mathsf{plus}(x, \mathsf{Succ}(y)) &= \mathsf{plus}(\mathsf{Succ}(x), y) \\
\mathsf{sumlist}(\mathsf{Nil}) &= \mathsf{Zero} \\
\mathsf{sumlist}(\mathsf{Cons}(x, xs)) &= \mathsf{plus}(x, \mathsf{sumlist}(xs)) \\
\mathsf{listsum}(\mathsf{Nil}) &= \mathsf{Zero} \\
\mathsf{listsum}(\mathsf{Cons}(x, \mathsf{Nil})) &= x \\
\mathsf{listsum}(\mathsf{Cons}(x, \mathsf{Cons}(y, xs))) &= \mathsf{listsum}(\mathsf{Cons}(\mathsf{plus}(x, y), xs))
\end{aligned}$$

Prove that the formula

$$\forall xs.\ \mathsf{listsum}(xs) =_{\mathsf{Nat}} \mathsf{sumlist}(xs) \tag{A}$$

is a theorem in the standard model by using induction and equational reasoning via $\rightsquigarrow$.

- Briefly state on which variable(s) you perform induction, and which induction scheme you are using.
- Write down each case explicitly and also write down the IH that you get, including quantifiers.
- Write down each single $\rightsquigarrow$-step in your proof.
- You will need one further auxiliary property (B). Write down this property and prove it in the same way as it is required for formula (A). Only exception: if you need further auxiliary properties for proving (B), then just state these properties without proving them.

**Exercise 4: Verification of Imperative Programs** 27

Consider the following program $P$ that computes the division of $x$ by $y$, i.e., the quotient $q$ and the remainder $r$ is computed such that $x = q \cdot y + r \wedge r < y$ should be satisfied.

```
q := 0;
while (x >= y) {
  q := q + 1;
  x := x - y;
}
r := x;
```

(a) Formulate pre- and post-conditions that state partial correctness of $P$. (3)

(b) Construct a proof tableau for proving partial correctness. (12)

```
q := 0;

while (x >= y) {

  q := q + 1;

  x := x - y;

}

r := x;
```

(c) Find a reasonable precondition that ensures termination and complete the proof tableau for proving termination formally. (12)

```
q = 0;

while (x >= y) {

  q := q + 1;

  x : = x - y;

}

(* the part after the while-loop should be omitted *)
```

Here is another blank template that can be used for a second attempt of either (b) or (c). If you use this template, please clearly indicate which of your solutions should (not) be graded.

```
q := 0;

while (x >= y) {

  q := q + 1;

  x := x - y;

}

r := x;
```