Lastname: ______________________________

Firstname: ______________________________

Matriculation Number: ______________________________

Exercise	Points	Score
Well-Definedness of Functional Programs	26	
Verification of Functional Programs	35	
Single Choice	6	
Verification of Imperative Programs	33	
$\sum$	100	

- The duration of 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 and use extra blank sheets if more space is required.
- Your answers can be written in English or German.

Exercise 1: Well-Definedness of Functional Programs 26

Consider the following functional program that computes the minimum of a list of natural numbers.

$$\begin{aligned}
\mathsf{data\ Nat} &= \mathsf{Zero} : \mathsf{Nat} && (1)\\
&\mid \mathsf{Succ} : \mathsf{Nat} \to \mathsf{Nat} && (2)\\
\mathsf{data\ List} &= \mathsf{Nil} : \mathsf{List} && (3)\\
&\mid \mathsf{Cons} : \mathsf{Nat} \times \mathsf{List} \to \mathsf{List} && (4)\\
\mathsf{min}(\mathsf{Succ}(x), \mathsf{Succ}(y)) &= \mathsf{Succ}(\mathsf{min}(y, x)) && (5)\\
\mathsf{min}(\mathsf{Zero}, y) &= \mathsf{Zero} && (6)\\
\mathsf{min}(x, \mathsf{Zero}) &= \mathsf{Zero} && (7)\\
\mathsf{minlist}(\mathsf{Cons}(x, \mathsf{Nil})) &= x && (8)\\
\mathsf{minlist}(\mathsf{Cons}(x, \mathsf{Cons}(y, xs))) &= \mathsf{minlist}(\mathsf{Cons}(\mathsf{min}(x, y), xs)) && (9)
\end{aligned}$$

(a) Is the program pattern disjoint and/or pattern complete? If not, then modify it in a sensible way such that it becomes both pattern disjoint and pattern complete and briefly explain your modifications. (8)

(b) First, compute all dependency pairs of the program. Second, indicate which of these pairs can be removed by the subterm-criterion and/or by the size-change principle with a brief justification why these pairs can be removed or why they cannot be removed. (8)

(c) Compute and write down the set of usable equations w.r.t. the dependency pairs of $\mathsf{minlist}^\sharp$. (10)

Afterwards, prove termination of minlist by a polynomial interpretation. You only have to provide the interpretations of the relevant function symbols, and do not have to argue why these interpretations satisfy the termination constraints.

$$\begin{aligned} p_{\mathsf{minlist}^\sharp}(xs) &= \dots \\ \dots &= \dots \end{aligned}$$

Exercise 2: Verification of Functional Programs 35

Consider the following functional program on lists over some element type E and Booleans, where $\mathsf{g} : \mathsf{E} \to \mathsf{Bool}$ is some defined function where the defining equations are not of interest for this exam question.

$$
\begin{aligned}
\mathsf{g}(\dots) &= \dots\\
\mathsf{if}(\mathsf{True}, xs, ys) &= xs\\
\mathsf{if}(\mathsf{False}, xs, ys) &= ys\\
\mathsf{filter}(\mathsf{Nil}) &= \mathsf{Nil}\\
\mathsf{filter}(\mathsf{Cons}(x, xs)) &= \mathsf{if}(\mathsf{g}(x), \mathsf{Cons}(x, \mathsf{filter}(xs)), \mathsf{filter}(xs))
\end{aligned}
$$

(a) Specify the property that any list xs is longer than $\mathsf{filter}(xs)$. To this end, you should define a recursive function $\mathsf{longer} : \mathsf{List} \times \mathsf{List} \to \mathsf{Bool}$ such that (5)

- longer checks that the first argument is at least as long as the second argument, and
- longer does not invoke any other auxiliary algorithm; in particular, there should not be any definition of length or $\mathsf{lessOrEqual}$ or similar functions.

and then write down the intended property with the help of longer.

(b) Provide a proof of the specified property by using induction and equational reasoning via $\rightsquigarrow$. (30)

- 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 any IH that you get, including quantifiers.
- Write down each single $\rightsquigarrow$-step in your proof.
- You will require a case-analysis within the inductive proof for the if-then-else, e.g., you might have to consider two cases $\mathsf{g}(x) = \mathsf{True}$ and $\mathsf{g}(x) = \mathsf{False}$.
- You will need at least one further auxiliary property, which is a monotonicity property of the longer-function of the form $\mathsf{longer}(\dots, \dots) \longrightarrow \mathsf{longer}(\dots, \dots)$. Write down this property and prove it in the same way in that you have to prove the main property.
  If you require further auxiliary properties, just state them without giving a proof.
- You may write just b instead of $b =_{\mathsf{Bool}} \mathsf{True}$ within your proofs.
- In case you did not solve part (a), you can ask the instructor for the solution to part (a) and continue here, but then part (a) will be graded with 0 points.

Exercise 3: 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. ____ Statement: A correctness proof via refinement roughly works as follows: one shows that an abstract algorithm satisfies some property P, and that a concrete algorithm is a correct implementation of the abstract algorithm in order to show that the concrete algorithm has property P.
2. ____ Assume there is some algorithm A computing function f with certificate generation. Further assume that there is a certificate checking algorithm C for function f, in particular supporting the certificates that are generated by A.

   Statement: If algorithm C has formally been verified then all certificates generated by A will be accepted by C.

Exercise 4: Verification of Imperative Programs 33

Consider the following program P. You can assume that array a has a length of n, i.e., $a = [a[0], ..., a[n-1]]$ and $n \geq 0$.

```
b := false;
i := 0;
while (i < n) {
  b := (b || (a[i] == x));
  i := i + 1;
}
```

(a) Figure out what P computes in variable b. Write this down informally, and also provide a specification in form of a Hoare triple. (3)

(b) Construct a proof tableau for proving termination. (10)
If required, it is allowed to add preconditions which are essential to ensure termination.
Clearly specify the variant e.

```
b := false;

i := 0;

while (i < n) {

  b := b || a[i] == x;

  i := i + 1;

}
```

(c) Prove partial correctness of the Hoare triple that you specified in the first part. (20)

```
b := false;

i := 0;

while (i < n) {

  b := b || a[i] == x;

  i := i + 1;

}
```