Prof. Dr. Jürgen Giesl

Jera Hensel

Bachelor/Master Exam

First Name: _____

Last Name: _____

Immatriculation Number: _____

Course of Studies (please mark exactly one):

- ○ **Informatik Bachelor**
- ○ **Mathematik Master**
- ○ **Other:** _____
- ○ **Informatik Master**
- ○ **SSE Master**

	Maximal Points	Achieved Points
Exercise 1	8	
Exercise 2	10	
Exercise 3	8	
Exercise 4	10	
Exercise 5	9	
Exercise 6	7	
Exercise 7	8	
Total	60	
Grade	-	

Instructions:

- On every sheet please give your **first name**, **last name**, and **immatriculation number**.
- You must solve the exam **without** consulting any **extra documents** (e.g., course notes).
- Make sure your answers are readable. Do not use **red or green pens or pencils**.
- You can solve the exercises in **English** or **German**.
- Please answer the exercises on the **exercise sheets**. If needed, also use the back sides of the exercise sheets.
- Answers on extra sheets can only be accepted if they are clearly marked with your name, your immatriculation number, and the **exercise number**.
- **Cross out** text that should not be considered in the evaluation.
- Students that try to cheat **do not pass** the exam.
- At the end of the exam, please return **all sheets together with the exercise sheets**.

Name: **Immatriculation Number:**

Exercise 1 (Theoretical Foundations): (5 + 3 = 8 points)

Let $\varphi = \mathrm{p}(0, \mathrm{s}^2(0)) \wedge \forall X, Y\, \exists U\, (\mathrm{p}(X, \mathrm{s}(Y)) \rightarrow \mathrm{p}(\mathrm{s}^2(X), U)) \wedge \neg \mathrm{p}(\mathrm{s}^2(0), 0)$ and $\psi = \exists Z\, \mathrm{p}(\mathrm{s}^2(0), Z)$ be formulas over the signature (Σ, Δ) with $\Sigma = \Sigma_0 \cup \Sigma_1, \Sigma_0 = \{0\}, \Sigma_1 = \{\mathrm{s}\}$, and $\Delta = \Delta_2 = \{\mathrm{p}\}$.

a) Prove that $\{\varphi\} \models \psi$ by means of SLD resolution.

Hint: First transform the formula $\varphi \wedge \neg\psi$ into a clause set that is satisfiable iff $\varphi \wedge \neg\psi$ is satisfiable.

b) Explicitly give a Herbrand model of the formula φ (i.e., specify a carrier and a meaning for all function and predicate symbols). You do not have to provide a proof for your answer.

Hint:

- Here, $\mathrm{s}^2(0)$ stands for $\mathrm{s}(\mathrm{s}(0))$ and $\mathrm{s}^2(X)$ stands for the term $\mathrm{s}(\mathrm{s}(X))$.

Name: **Immatriculation Number:**

Exercise 2 (Procedural Semantics, SLD tree): (4 + 6 = 10 points)

Consider the following Prolog program $\mathcal{P}$ which, for each pair of adjacent numbers in a list, swaps the numbers if the first number is greater than the second number.

```
bubble([],[]).
bubble([X,Y|XS],[Y|YS]) :- X > Y, !, bubble([X|XS],YS).
bubble([X|XS],[X|YS]) :- bubble(XS,YS).
```

a) The program $\mathcal{P}'$ results from $\mathcal{P}$ by **removing the cut**. Consider the following query:

```
?- bubble([5,6,2],[5,X,Y]).
```

For the logic program $\mathcal{P}'$ (i.e., **without the cut**), show a successful computation for the query above (i.e., a computation of the form $(G, \varnothing) \vdash^{+}_{\mathcal{P}'} (\square, \sigma)$ where $G = \{\neg$`bubble([5,6,2],[5,X,Y])`$\}$). You may leave out the negations in the queries and you do not have to show how the substitutions operate on variables that only occur in (renamed) program clauses.

b) Please give a graphical representation of the SLD tree for the query

```
?- bubble([3,1,4],Res).
```

in the program $\mathcal{P}$ (i.e., **with the cut**). For every part of a tree that is cut off by evaluating `!`, please indicate the cut by crossing out the edge that leads to the cut-off part. For the cut-off parts only indicate the first cut-off goal, but do not evaluate further. Please also indicate all answer substitutions.

Exercise 3 (Definite Logic Programming): (8 points)

We consider the following problem: Persons of different weights should be distributed onto boats with a capacity of 200 kg each. Here, we assume an infinite number of boats.
Implement a predicate `dist/2` in Prolog. If the first argument is a list of persons constructed using the function `person(`w`)` for a weight $w \in \mathbb{N}$, the predicate should return a list of boats as its second argument, where a boat is constructed using the function `boat(`*list*`)` for a list of persons *list*.
Assume that the persons are lined up and the boats are filled in a **greedy** way: The first boat is filled until the next person in line does not fit in (i.e., the sum of the weights of the people in the boat and the weight of the next person in the line exceeds 200 kg). Then, this person is the first to enter the next boat, etc. However, you should not use a new boat if the next person in line fits into the currently filled boat.
For example, the query `?- dist([person(140),person(70),person(70)], Boats)` should succeed with the only answer `Boats = [boat([person(140)]), boat([person(70),person(70)])]`. The query `?- dist([person(70),person(140),person(70)], Boats)` should succeed with the only answer `Boats = [boat([person(70)]), boat([person(140)]), boat([person(70)])]`.
You **may only use** the built-in predicates `is/2`, `</2`, `=</2`, the cut, and the usual arithmetic operators such as `+`, `*`, `-`, `/`. You may assume that no person in the line weighs more than 200 kg.

Exercise 4 (Fixpoint Semantics): **(5 + 2 + 3 = 10 points)**

Consider the following logic program $\mathcal{P}$ over the signature (Σ, Δ) with $\Sigma = \{0, s\}$ and $\Delta = \{p\}$.

```
p(0, X, X).
p(s(X), s(s(Y)), s(Z)) :- p(X, Y, Z).
```

a) For each $n \in \mathbb{N}$ explicitly give $\underline{\text{trans}}^n_{\mathcal{P}}(\varnothing)$ in closed form, i.e., using a non-recursive definition.

b) Compute the set $\text{lfp}(\underline{\text{trans}}_{\mathcal{P}})$.

c) Give $F[\![\mathcal{P}, \{\neg \texttt{p(X, s(s(0)), Z)}\}]\!]$.

Exercise 5 (Meta-Programming): (9 points)

Implement a predicate `choose/2` which manipulates the term given as its first argument in such a way that for every subterm that is a compound of the form $f(t_1, \ldots, t_n)$ with $n \geq 2$, all arguments except one are removed (i.e., every subterm of the form $f(t_1, \ldots, t_n)$ is replaced by $f(t_i)$ for some $1 \leq i \leq n$). All other subterms should not be modified. For example, the query `?- choose(p(a,q(b,X)),R)` should succeed with the answers `R = p(a)`, `R = p(q(b))`, and `R = p(q(X))`.
You **may only use** the built-in predicates `var/1`, `atomic/1`, `compound/1`, and `=../2`.

Exercise 6 (Universality): (7 points)

Consider a function $f : \mathbb{N} \to \mathbb{N}$. The function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined as:

$$g(x, y) = k \text{ iff } f(x) + f(y) = f(k),\ f(x),\ f(y), \text{ and } f(k) \text{ are defined, and}$$
$$\text{for all } 0 \leq k' < k, \text{ we have } f(x) + f(y) \neq f(k') \text{ and } f(k') \text{ is defined.}$$

As an example, consider the function $\hat{f} : \mathbb{N} \to \mathbb{N}$ with $\hat{f}(x) = |x - 4|$ (i.e., the absolute value of $x - 4$). The function $\hat{g} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, constructed as described above, computes $\hat{g}(6, 3) = 1$. The reason is that for $x = 6, y = 3$, 1 is the smallest k such that $\hat{f}(x) + \hat{f}(y) = \hat{f}(k)$ and $\hat{f}(x) + \hat{f}(y) \neq \hat{f}(k')$ for all $0 \leq k' < k$. Indeed, $\hat{f}(6) + \hat{f}(3) = 2 + 1 = 3 = \hat{f}(1) = \hat{f}(k)$ and $\hat{f}(0) = 4 \neq 3$.

Consider a definite logic program $\mathcal{P}$ which computes the function f using a predicate symbol $\underline{\mathtt{f}} \in \Delta^2$:

$$f(x) = k \text{ iff } \mathcal{P} \models \underline{\mathtt{f}}(\underline{x}, \underline{k}).$$

Here, numbers are represented by terms built from $\mathtt{0} \in \Sigma_0$ and $\mathtt{s} \in \Sigma_1$ (i.e., $\underline{0} = \mathtt{0}, \underline{1} = \mathtt{s(0)}, \underline{2} = \mathtt{s(s(0))}, \ldots$).

Please extend the definite logic program $\mathcal{P}$ such that it also computes the function g using the predicate symbol $\underline{\mathtt{g}} \in \Delta^3$ (but without any built-in predicates):

$$g(x, y) = k \text{ iff } \mathcal{P} \models \underline{\mathtt{g}}(\underline{x}, \underline{y}, \underline{k}).$$

Hints:

- You may use the predicates `plus/3` and `ne/2` (for not equal):

```
plus(X,0,X).
plus(X,s(Y),s(Z)) :- plus(X,Y,Z).

ne(0,s(Y)).
ne(s(X),0).
ne(s(X),s(Y)) :- ne(X,Y).
```

Exercise 7 (Programming with CLP): (8 points)

In this task, we use Prolog to solve simple addition equations where single digits may be replaced by variables. Here, a number is represented as the list of its digits. For example, the list $[2, 1, 7]$ represents the number 217.
Implement a Prolog predicate `add/3` such that `add(`t_1`,`t_2`,`t_3`)` is true if and only if t_1, t_2, and t_3 are lists of integers between 0 and 9 of the same length n and the sum of the numbers represented by t_1 and t_2 is the number represented by t_3. For example, the query `?- add([X,4,6],[1,7,1],[2,Y,Z])` should succeed with the unique substitution `X = 0, Y = 1, Z = 7`.

The following line is already given:

```
:- use_module(library(clpfd)).
```

Hint:

- For any two natural numbers with n digits, their sum is a natural number with $n + 1$ digits where the most significant digit is either 0 or 1 (i.e., it corresponds to a carry bit). For example, $523 + 674 = 1197$ and $323 + 674 = 0997$. Thus, it could be helpful to implement an auxiliary predicate `add/4` where `add(`t_1`,`t_2`,`t_3`,`c`)` holds if the sum of the numbers represented by t_1 and t_2 is the number represented by `[`c`|`t_3`]` where $c \in \{0, 1\}$.