Prof. Dr. Jürgen Giesl
Cornelius Aschermann, Jera Hensel

Bachelor/Master Exam Version V3B

First Name: ______________________________

Last Name: ______________________________

Immatriculation Number: ______________________________

Course of Studies (please mark exactly one):

- ○ **Informatik Bachelor**
- ○ **Mathematik Master**
- ○ **TK Master**
- ○ **Other:** ____________________

	Maximal Points	Achieved Points
Exercise 1	13	
Exercise 2	10	
Exercise 3	11	
Exercise 4	14	
Exercise 5	5	
Exercise 6	7	
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**.
- 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**.

Exercise 1 (Theoretical Foundations): **(4 + 4 + 5 = 13 points)**

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

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

Hint: First transform the formula $\varphi \wedge \neg\psi$ into an equivalent clause set.

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.

c) Prove or disprove: If $\mathcal{K}$ is a set of clauses without variables, S is a model of $\mathcal{K}$, $K_1, K_2 \in \mathcal{K}$ and R is a resolvent of K_1 and K_2, then S is a model of $\mathcal{K} \cup \{R\}$.

Exercise 2 (Procedural Semantics, SLD tree): (5 + 5 = 10 points)

Consider the following Prolog program $\mathcal{P}$ which can be used to check whether a list contains 4 or 6, but it does not contain any 2 before the first 4 or 6.

```
e(2).
e(4).
e(6).
p([X|_]):- e(X),!,not(X = 2).
p([_|XS]):- p(XS).
not(X):- X,!,fail.
not(_).
```

As an example, the query `p([1,2,4,8])` would not be provable (since it contains a 2 and there is no 4 or 6 before).

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

```
?- p([1,2,4,8]).
```

For the logic program $\mathcal{P}'$ (i.e., **without the cuts**), please show a successful computation for the query above (i.e., a computation of the form $(G, \varnothing) \vdash^{+}_{\mathcal{P}'} (\Box, \sigma)$ where $G = \{\neg \texttt{p[1,2,4,8]}\}$). You may leave out the negations in the queries.

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

```
?- p([1,4]).
```

in the program $\mathcal{P}$ (i.e., **with the cuts**). For every part of a tree that is cut off by evaluating `!`, please indicate the cut by marking the corresponding edge. For the cut-off parts only indicate the first cut-off goal, but do not evaluate further.

Exercise 3 (Fixpoint Semantics): (5 + 3 + 3 = 11 points)

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

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

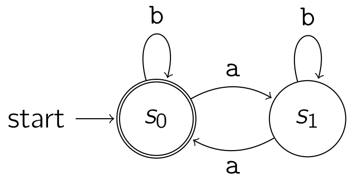
a) For each $n \in \mathbb{N}$ explicitly give $\underline{\text{trans}}_{\mathcal{P}}^{n}(\varnothing)$ in closed form, i.e., using a non-recursive definition.

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

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

Exercise 4 (Definite Logic Programming): (7 + 7 = 14 points)

a) We consider Deterministic Finite Automata (DFAs). An example for such an automaton is given below. It accepts all words where the number of "a" characters in the word is even.



We encode this automaton into Prolog facts as follows:

```
start(s0).
final(s0).
delta(s0,a,s1).
delta(s1,a,s0).
delta(s1,b,s1).
delta(s0,b,s0).
```

As a quick reminder: A DFA is a five-tuple $(Q, \Sigma, \delta, q_0, F)$. Here, Q is a set of states (in our case $\{s_0, s_1\}$), Σ is the set of alphabet symbols (in our case $\{\mathtt{a}, \mathtt{b}\}$). The transition function δ: $Q \times \Sigma \mapsto Q$ maps the current state to the next state given that a certain symbol from Σ was read. The automaton starts in the start state q_0 and accepts the word if it stops in a final state from the set $F \subseteq Q$ (in our case $F = \{s_0\}$).

We say that an automaton $(Q, \Sigma, \delta, q_0, F)$ accepts a word $w = (a_1, a_2, \ldots, a_n) \in \Sigma^n$ if there is a run $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots \xrightarrow{a_n} q_n$ such that for all $i \in \{1 \ldots, n\}$ it holds that $\delta(q_{i-1}, a_i) = q_i$ and $q_n \in F$.

In the example above, we encoded the start state q_0 with the fact `start(s0)`, the set of final states F is encoded by the fact `final(s0)`, the transition function is encoded by the `delta/3` predicate such that `delta(`q_i`, `a`, `q_j`)` holds iff $\delta(q_i, a) = q_j$. The sets Q and Σ are implicitly defined by the arguments of `delta`.

Implement a predicate `accepts/1`. The query: `?- accepts(Word)` should succeed iff the DFA accepts the given word. In our example, the query `?- accepts([a,b,a])` should succeed but the query `?- accepts([a,b])` should fail. Your clause for `accepts` should work for any DFA (i.e., for any clauses defining `start`, `final`, and `delta`).

b) Consider the set partition problem: Given a set $S = \{a_1, \ldots, a_n\}$ of integer numbers, find a partition of S into two sets L and R such that

- $\sum_{a_i \in L} a_i = \sum_{a_i \in R} a_i$
- $L \cup R = S$
- $L \cap R = \varnothing$.

Implement a predicate `partition/3` such that `?- partition(S,L,R)` succeeds iff `L` and `R` are a valid partition of `S`. For example, `partition([1,2,3],L,R)` should succeed with answer substitution `L = [1,2], R = [3]`. On lists with duplicate entries your implementation may behave arbitrarily.

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

Consider the well known function $fib(i)$ that returns the i-th Fibonacci number. A possible Prolog implementation for fib is:

```
fib(0,0):- !.
fib(1,1):- !.
fib(X,Y):-
    XPP is X-2, fib(XPP,YPP),
    XP is X-1, fib(XP, YP),
    Y is YP + YPP,!.
```

However, this implementation has roughly exponential runtime. There are many ways to improve this. In this exercise we will use memoization to improve the performance drastically. Implement a predicate `mem(X)` that will try to prove `X`. If the proof of `X` succeeds, then the proof of `mem(X)` is also successful. Moreover, if the resulting answer substitution σ maps `X` to a ground term, then σ(`X`) is added to the database of program clauses.
Given this predicate, the following definition of `fib` should only require a linear number of evaluations of `fib`.

```
:-dynamic(fib/2).

fib(0,0):- !.
fib(1,1):- !.
fib(X,Y):-
    XPP is X-2, mem(fib(XPP,YPP)),
    XP is X-1, mem(fib(XP, YP)),
    Y is YP + YPP,!.
```

Hints:

- You may use the built-in predicates `assertz/1`, `asserta/1`, `ground/1` and the cut operator.

Exercise 6 (Difference Lists): (7 points)

Consider the following logic program $\mathcal{P}$.

```
append_diff(A-B,B-C,A-C).

r([1,2,3,4|X]-X).
r(Z):- r([H|T]-X), append_diff(T-X,[H|Y]-Y,Z).
```

Explicitly give the set of all answer substitutions for the query `?- r(ZS)` (up to variable renaming).