Kiel University
Department of Computer Science
Programming Languages and Compiler Construction

Priv.-Doz. Dr. Frank Huch, Sandra Dylus, B.Sc. Jan-Hendrik Matthes, Rajib Chandra Das

Institut für Informatik

3. Exam on „Advanced Programming (pre-masters)“

SS 19

You can obtain 28 points within two assignments. To pass the test, you have to reach at least 14 points.

For the test, you can work for 90 minutes. It is not allowed to use any material other than a pen. Mobile phones have to be turned off.

Hold your Student ID Card present, we will check it during the examination.

You will be informed about the results on Thursday, July 18, 2019 from 11:30 AM to 12PM in CAP4 - R.715.

Exercise 1 - Quiz

12 Points

Multiple Choice

Mark all correct answers with an X. **If you want to change your answer after marking a statement, fill the original square and draw a new one, as shown in the example.** Note that any number of answers can be correct. Each question is worth 1.5 points. For each incorrect answer 0.5 points are deducted; negative scores for a question are not possible.

Example

1. Which cities are in Germany?
   - ☒ Kiel
   - ☒ Hamburg
   - ☐ London
   - ■ ☐ Paris
   - ☒ Berlin

Questions

1. Which of the following statements about the unification algorithm are true?
   - ☐ The algorithm does not terminate if no *mgu* exists.
   - ☐ If the two input terms are not unifiable, an empty substitution is returned.
   - ☐ The occurs check avoids non-termination.
   - ☐ The algorithm always terminates.
   - ☐ Unification is used in Haskell's type inference algorithm.
2. Let ϕ be an *mgu* for the terms t_1 and t_2. Which of the following statements hold?
   - ☐ $\text{ds}(\phi(t_1), t_2) = \{\}$
   - ☐ $t_1 = \phi(t_2)$
   - ☐ $\phi(t_1) = \phi(\phi(t_2))$
   - ☐ $t_1 \neq t_2$
   - ☐ $(\phi \circ \phi)(t_1) = \phi(t_2)$
3. Which of the following patterns unify with the expression `[42, true]`?
   - ☐ `[X | [true]]`
   - ☐ `[42 | true]`
   - ☐ `X`
   - ☐ `[42, False | X]`
   - ☐ `[[42] | X]`
4. Which of the following sentences are correct?
   - ☐ Prolog uses the selection strategy FIRST in the SLD resolution.
   - ☐ $\text{ds}(f(a, g(Y), 73), f(X, b, 73)) = \{X, a, g(Y), b\}$
   - ☐ `?- g(X, Xs) = X.` has no solution in SWI-Prolog.
   - ☐ σ is a most general unifier, if for all unifiers σ' there exists a substitution ϕ such that $\sigma' = \phi \circ \sigma$.
   - ☐ `?- findall(X, member(X, [21, 42, 73]), L).` has more than one solution.

5. Which of the following queries are answered with `true` or a binding for the occuring variables.

   ☐ `?- 42 + 31 is 31 + 42.`

   ☐ `?- 42 is 7 * 6.`

   ☐ `?- 20 + 1 = 22 - 1.`

   ☐ `?- X = 42 + 31.`

   ☐ `?- 21 * 2.`

Answer the following questions directly on this sheet of paper.

1. *(2 points)* Give all solutions of the goal `?- p(Y).`.

```
1 p(X) :- q(X), q(a), !.
2 p(X) :- q(X), !, q(c).
3 p(b).
4
5 q(b) :- q(a).
6 q(c).
```

2. *(1 point)* Give all solutions of the goal `?- append([1|Xs], [3|Ys], [1, 2, 3, 4, 1, 3]).`.

3. *(1.5 points)* Give the most general unifier (if it exists) for the terms `fun(g(Y, Z), [Y|[h(X)|[]]])` and `fun(X, [42|Xs])`. Otherwise, explain why no *mgu* exists.

Exercise 2 - Programming in Prolog 16 Points

In this excercise we use Peano numbers. The structure should be clear from the following predicate, which can be used to check whether a given value is a Peano number.

```
1 isPeano(o).
2 isPeano(s(N)) :- isPeano(N).
```

1. *(1 point)* Define a predicate `gt(N, M)` that checks whether the Peano number `N` is greater than the Peano number `M`.
2. *(4 points)* Define a predicate `fromTo(N, M, Xs)` that computes the list `Xs` with all Peano numbers between the Peano numbers `N` and `M`. For example `?- fromTo(o, s(s(o)), Xs).` should compute the result `Xs = [o, s(o), s(s(o))]`.
3. *(3 points)* Define a predicate `dropLess(N, Xs, Ys)` that removes all Peano numbers (that are less than `N`) in the beginning of `Xs` until an element is found that is greater than or equal to `N`. The remaining list should then be equal to `Ys`. Implement this predicate without using a helper predicate (e.g. `leq`) from the lecture or exercise sheets. The only helper predicate you are allowed to use is `gt` (from part 1).
4. *(8 points)* We consider the following Prolog program

```
1 nth(o,    [X|_],  X).
2 nth(s(N), [_|Xs], X) :- nth(N, Xs, X).
```

and the goal `?- nth(N, [s(o), s(s(Y))], X), s(N) = X.`.

Construct the SLD tree for this goal **according to the format introduced in the lectures and exercises**. For simplicity, replace the underscores (`_`) in the rules for `nth` by a variable `U`.