Instructions for Students

- Write your name and student number on the exam sheet and on every solution sheet you hand in and also sign them.
- This is a closed book exam: the only resources allowed are blank paper and pens (do not use pencils).
- Write neatly and clearly. The clarity of your explanations will affect your grade.
- The duration of the exam is 2 hours.

Good luck!

Do not write in this space

<table>
<thead>
<tr>
<th>Question</th>
<th>Marks</th>
<th>Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Exercise 1 (24 marks)

a. (4 marks) Briefly describe the main differences, advantages and disadvantages of static typing and dynamic typing.

b. (4 marks) Briefly describe the concept of tail recursion, and why it is desirable to write tail recursive functions.

c. (4 marks) Briefly describe the concept of mixins in Ruby.

d. (4 marks) The box model of Prolog execution is a simple way to show the control flow. Briefly sketch and describe the box model.

e. (4 marks) Erlang supports higher-order functions. Briefly explain this concept and give an example on how to use such functions.

f. (4 marks) What is the result of the following Haskell expression?
\[(x,y) \mid x \leftarrow [1..4], y \leftarrow [1..4]\]

Exercise 2 (8 marks) Write a Ruby function timeseries(a), which takes as argument a time series (i.e., sequence of values) stored in an array and prints the minimum, the maximum and the average over the time series data. For instance, with a = [1, 6, 7, 3, 3] the function prints out min = 1, max = 7, avg = 4.

Exercise 3 (8 marks) Extend the Ruby class Fixnum with a method square_root_times that, if called for a number n and a code snippet, executes the code snippet \(\lceil\sqrt{n}\rceil\) times. For example, 5.square_root_times{ puts 'hello world!' } produces

hello world!
hello world!
hello world!

You are not allowed to use a built-in Ruby function to compute the square root of numbers.

Exercise 4 (16 marks) Write a Prolog program rmdup(X,Y) that removes all duplicates in list X and returns in Y the duplicate-free list. For instance, given the list X = [a,b,a,c,a,a,b], the duplicate-free list is Y = [c,a,b]. The order of the elements in Y does not matter. Hint: Implement first a function member(E,L) to check whether E is a member in L and use it in rmdup (that is, you have to implement the built-in predicate member/2) yourself).
Exercise 5 (8 marks) A well-known strategy for writing Prolog programs is “generate and test”. That means, that one part of the program generates potential solutions, while another part of the program tests whether a potential solution is correct or not.

The predicate `between(L, U, X)` generates all integers between L and U. For example, calling `between(0, 3, X)` generates the following output:

\[
X = 0 ; \\
X = 1 ; \\
X = 2 ; \\
X = 3 .
\]

Using the `between(L, U, X)` generator, write a Prolog rule `square(S)` that tests whether a number S is the square of an integer (for example, `square(4)` would return true, `square(3)` would return false).

Exercise 6 (12 marks) Write an Erlang server that computes the Euclidean distance between two points in the plane, i.e., when it receives two coordinates, it computes the distance between the points and sends it back. The server process receives a tuple containing the ID of the process requesting the information and two coordinates. For example, a request may look like this: `{PID1, 3.5, 4.8, 2.0, 7.3}`, where PID1 is the ID of the requesting process, while \( x_1 = 3.5, y_1 = 4.8 \) are the coordinates of the first point and \( x_2 = 2.0, y_2 = 7.3 \) are those of the second one. The Euclidean distance is computed as follows: \( \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \). Assume that you have access to a function `sqrt` that computes the square root of a number.

Show also how the server is started.

Exercise 7 (14 marks) Write a Haskell module that exports a function `subseq` that takes as input a list x and a list y and returns true if x is a subsequence of y and false otherwise. For instance, `subseq [1,2,3] [3,4,1,2,3,5]` returns true, whereas `subseq [1,2,3] [1,2]` returns false.

Exercise 8 (10 marks) Write a Haskell module that exports a function `noOfElem` that takes as input an element x and a list and returns the number of occurrences of x in the list. For instance, `NoOfElem 1 [1,2,3,1]` returns 2.
Solution 1

a. Static typing:
   - types and their constraints are checked before executing a program
   - pro: less error-prone
   - con: sometimes too restrictive

Dynamic typing:
   - type checking is done during program execution
   - pro: more flexible
   - con: more difficult to debug

b. A function is tail-recursive if there is no operation after the recursive call, that is, no operations are executed after the recursive call terminates. As a consequence, no data need to be stored on the stack that is needed when the recursive call terminates. Hence, different from non-tail-recursive function, for tail-recursive functions the stack does not grow with each recursive call.

c. A mixin in Ruby is a combination of modules and classes. More specifically, a module can be included in a class definition. By doing so, all methods of the module are added and available to the class. Mixins have some similarity to the concept of multiple inheritance.

d. The box model provides a simple way to show the control flow of a Prolog program. A box represents the invocation of a single predicate. The box has four ports (with associated events):
   - CALL: The first call of a predicate; control enters into the box
   - EXIT: The goal has been proven
   - REDO: The system comes back to a goal, trying to re-satisfy it, i.e., backtracking
   - FAIL: The goal/predicate fails

![Box Model Diagram]


e. A higher-order function is a function that accepts another function as input or returns a function in output. For example, the following expression increments the elements of a list by 1 using an anonymous function:
   ```erlang
   lists:map(fun(X) -> X+1 end, [1,2,3,4]) returns [2,3,4,5]
   ```
f. [(1,1), (1,2), (1,3), (1,4), (2,1), (2,2), ..., (4,4)]

Solution 2

```ruby
def timeseries( a )
    min = max = a[0]
    sum = 0
    a.each{ |v|
        max = v if v > max
        min = v if v < min
        sum += v
    }
    puts "min = #{min}"
    puts "max = #{max}"
    puts "avg = #{sum/a.length}" 
end
```

Solution 3

```ruby
class Fixnum
    def square_root_times
        i = 0
        while i * i < self
            i += 1
            yield
        end
    end
end
```

Solution 4

```prolog
member( X, [X|_] ).
member( X, [_|Xs] ) :- member(X, Xs).

rmdup( [], [] ).
rmdup( [H|T], R):-
    member( H, T ),
    !,
    rmdup(T,R).
rmdup( [H|T], [H|Rest] ):-
    rmdup( T, Rest ).
```
Solution 5

\[
\text{square}( S ) :- \\
\quad \text{between}( 0, S, X ), \\
\quad S \text{ is } X * X, \\
\quad !. 
\]

Solution 6

\-module(euclidserver). 
\-export([loop/0]). 

loop() -> 
\receive 
\quad \{PID,X1,Y1,X2,Y2\} \rightarrow 
\quad XD = X1 - X2, 
\quad YD = Y1 - Y2, 
\quad PID ! \text{sqrt}(XD * XD + YD * YD), 
\quad loop() 
\end. 

\text{Pid} = \text{spawn(fun euclidserver:loop/0)}. 

Solution 7

module Subseq (  
    subseq  
) where

-- Verify whether x occurs at the beginning of y  
sseq :: Eq a => [a] -> [a] -> Bool  
sseq x [] = False  
sseq [] y = True  
sseq (x:xs) (y:ys) = if x == y then sseq xs ys else False

-- Verify whether x occurs somewhere in y  
subseq :: Eq a => [a] -> [a] -> Bool  
subseq x [] = False  
subseq x y =  
    if sseq x y then  
        True  
    else  
        subseq x (tail y)

Solution 8

module NoOfElem (noOfElem) where

noOfElem :: Eq a => a -> [a] -> Int  
noOfElem x [] = 0  
noOfElem x (h:t) = (if h == x then 1 else 0) + noOfElem x t