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>Exercise</th>
<th>Marks</th>
<th>Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</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 (20 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 mixins in Ruby.

c. (4 marks) What is the following Prolog program doing?

\[
\text{foo} :- \\
\quad \text{repeat,} \\
\quad \text{read}(X), \\
\quad \text{write}(X), \\
\quad \text{nl}, \\
\quad X = 'quit', \\
\quad !.
\]

d. (4 marks) What is the following Haskell list comprehension producing?

\[
[(4-x,y) \mid (x,y) \leftarrow [(1,2), (2,3), (3,1)]]
\]

e. (4 marks) The following figure shows a hierarchy of linked Erlang processes, also called supervision tree. What happens if process I crashes?
Exercise 2 (12 marks) Write a Ruby function that implements counting sort for sorting an array of numbers, which works as follows: Given an array \( A \) of positive integers \( n_1, \ldots, n_k \in [0,k] \), create an array \( A' \) of size \( k+1 \). Each element of \( A \) is associated with an index in \( A' \). The algorithm counts then the number of occurrences of each element in \( A \) and stores it in the corresponding cell in \( A' \). Finally, with a scan of \( A' \) the elements can be retrieved in ascending (or descending) order.

For example, for the input array \( A = [6,5,1,7,8,1,2] \) we obtain \( A' = [0,2,1,0,0,1,1,1,1] \). The element \( A'[0] \) represents that the number 0 occurs zero times, \( A'[1] \) represents that the number 1 occurs two times, etc. Scanning \( A' \) allows to retrieve the sorted array \( [1,1,2,5,6,7,8] \).

Exercise 3 (8 marks) The following Ruby function computes \( \lceil \log_2 n \rceil \) recursively:

```ruby
def log( n )
  return 0 if n == 1
  if n.even?
    n = n / 2
    return log( n ) + 1
  else
    n += 1
    return log( n )
  end
end
```

Rewrite this function into a tail-recursive function.

Exercise 4 (10 marks) Write a Prolog program \( \text{pythagoras}(A,B,C) \) using the “generate and test” pattern, which for a given value \( C \) computes all possible integer values of \( A \) and \( B \) for which the theorem of Pythagoras holds, i.e., \( A^2 + B^2 = C^2 \). For instance, \( \text{pythagoras}(A,B,5) \) returns \( A=3, B=4 \) and \( A=4, B=3 \), whereas \( \text{pythagoras}(A,B,6) \) fails.

Hint: You can use the predicate \( \text{between}(L,U,X) \), which generates all integers between \( L \) and \( U \), e.g., \( \text{between}(0,2,X) \) generates

\[
X = 0 ; \\
X = 1 ; \\
X = 2 .
\]
**Exercise 5** (16 marks) Write a program in Prolog that goes through a list of numbers and selects numbers (starting from the beginning of the list) whose sum is smaller than a given capacity. So as long as there is still enough capacity left, the program keeps selecting numbers (skipping numbers that are too large). The program should return the result in a list. For example, given the list [2,5,3,8,1,12] and the capacity 14, the program should return the list [2,5,3,1] as this sums up to 11, which is less than 14 (when 8 and 12 are reached, they are too large to be included). The order of the items in the result does not matter.

**Exercise 6** (14 marks) Write a function `isbalanced` in Haskell that checks whether a string containing open and closed parentheses is balanced. A string of parentheses is balanced when every open one has a corresponding closed one and at any point there are not more closed ones than open ones. For example, the strings "", "(()" and "((()))" are balanced, whereas the strings "(()(" and "())" are not balanced.

**Exercise 7** (10 marks) Look at the following recursive Haskell program:

```haskell
mystery :: [a] -> Integer
mystery [] = 1
mystery (h:t) = 2 * (mystery t)
```

a. (4 marks) Briefly describe what the program does.

b. (6 marks) Transform the program into a tail-recursive one.

**Exercise 8** (10 marks) Write a function `loop` for an Erlang process that receives messages consisting of a single parameter and does the following:

- if the parameter is a number, it outputs to the console whether the number is positive, negative, or zero;
- if the parameter is “bye”, the process terminates;
- otherwise, an error message is printed, e.g., “Unexpected message”.

Show also how to start the process. (Hint: you can use a function `is_number(N)`, which is true if `N` is a number, and false otherwise)