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>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
<td></td>
</tr>
</tbody>
</table>
Exercise 1 (20 marks)

a. (4 marks) Briefly describe the concept of abstract data types and the advantages they introduced with respect to imperative/procedural programming.

b. (4 marks) Briefly describe the concept of a symbol in Ruby and its difference to strings.

c. (4 marks) What is the key difference in a Prolog query between using variables or atoms as parameters (e.g., between likes(gromit,cheese) and likes(X,Y))?

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

\[ [(x,y) \mid x \leftarrow [1..3], y \leftarrow [2..4], x /= y] \]

e. (4 marks) When you move the execution of an Erlang program from a single-processor machine to a multi-core machine or a cluster of computers, do you have to rewrite or adapt your program? Explain your answer.

Exercise 2 (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,

\[ 5 \text{.square_root_times} \{ \text{puts 'hello world!' } \} \]

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 3 (12 marks) Assume temperature data stored in an array \( t \). Write a Ruby function `max_period(t,x)` that calculates the length of the longest (hot) period, where all temperature values are greater than \( x \). The result is printed to the console. For example, for \( t = [20, 25, 26, 23, 27] \) and \( x = 24 \), the function prints The longest period greater than 24 is of length 2.

Exercise 4 (14 marks) Write a Prolog predicate `decode(L1, L2)` that decodes a run-length encoded list \( L1 \) into the uncompressed list \( L2 \). Assume a simple run-length encoding, where a sequence of \( N \) characters \( X \) is represented as a pair \( (N,X) \); sequences of length 1 are also encoded. For instance, the call `decode([ (2,a), (1,b), (3,c) ], L2)` should instantiate \( L2 = [a, a, b, c, c, c] \).
Exercise 5 (10 marks) Write a Prolog program \texttt{pythagoras(A,B,C)} using the “generate and test” pattern, which for a given value \(C\) computes all possible values of \(A\) and \(B\) for which the theorem of Pythagoras holds, i.e., \(A^2 + B^2 = C^2\) (\(A, B\) and \(C\) are integer numbers). For instance, \texttt{pythagoras(A,B,5)} returns \(A=3, B=4\) and \(A=4, B=3\), whereas \texttt{pythagoras(A,B,6)} fails.

Hint: You can use the predicate \texttt{between(L,U,X)}}, which generates all integers between \(L\) and \(U\), e.g., \texttt{between(0,3,X)} generates

\begin{verbatim}
X = 0 ;
X = 1 ;
X = 2 ;
X = 3.
\end{verbatim}

Exercise 6 (8 marks) Write a Haskell function \texttt{rotate} that rotates a list by \(n\), \(\geq 0\), places to the left. Here are a few examples:

\begin{verbatim}
rotate \[1,2,3,4,5\] 0 ⇒ \[1,2,3,4,5\]
rotate \[1,2,3,4,5\] 2 ⇒ \[3,4,5,1,2\]
rotate \[1,2,3,4,5\] 7 ⇒ \[3,4,5,1,2\]
\end{verbatim}

Exercise 7 (14 marks) Write a Haskell module that exports a function \texttt{split} that splits a list \(l\) at a given position \(n\) into two lists. The list and the split position are given as input parameter. The function returns a pair \((l_1, l_2)\) consisting of the two parts of the list; if \(n \leq 0\), \(l_1\) is empty and \(l_2\) contains the input list; if \(n\) is greater than the length of the input list, \(l_1\) contains the input list and \(l_2\) is empty. Here are a few examples:

\begin{verbatim}
split \[7,8,9,3,4\] 2 ⇒ \([7,8], [9,3,4]\)
split \[7,8,9,3,4\] -1 ⇒ \([], [7,8,9,3,4]\)
split \[7,8,9,3,4\] 7 ⇒ \([7,8,9,3,4], []\)
\end{verbatim}

Exercise 8 (14 marks) Write an Erlang module that implements and exports a server to count positive and negative votes of a ballot. If the server receives the message “yes”, the counter of the positive votes is incremented by 1, whereas the message “no” increments the number of negative votes by 1. If the message “info” is received, the server sends a message containing the number of positive and negative votes to a process that is registered with an atom \texttt{observer}, and it displays the counters on the screen. The message “Bye” terminates the process. Show also the command to start the server from the command line.
Solution 1

a. The procedural approach in imperative programming was taken further by introducing abstract data types (ADT). In ADTs, everything related to a type is encapsulated in one bundle, most importantly data itself and operations on the data. This is also known as information hiding and has several advantages: data can only be accessed via a specified operations/interface; the actual representation/implementation is hidden and can easily be changed/replaced without affecting the rest of the program; the code becomes more portable.

b. A symbols in Ruby is an identifier preceeded by a colon, e.g., :thisisasymbol. A symbol is a unique, immutable string, which cannot be changed. Whenever we use a symbol, it refers to the same object, whereas two identical strings are different objects.

c. A query with atoms as parameters answers a yes/no query, i.e., verifies whether the query statement is true or false in the knowledge base. A query with variables as parameters retrieves all instantiations (values) of the variables for which the query is true in the knowledge base.

d. [(1,2),(1,3),(1,4),(2,3),(2,4),(3,2),(3,4)]

e. No, Erlang programs don’t have to be modified to run on a multi-core. The main reason is that processes do not share any resources and communicate exclusively by message passing. Therefore, it makes no difference whether they run on the same machine or on different machines. The Erlang virtual machine will automatically adapt and use the underlying hardware.

Solution 2

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

Solution 3

def maxperiod( t, x )
  lmax = 0
i = 0
while i < t.length
    if t[i] > x
        l = 1
        l += 1 while i + l < t.length && t[i+l] > x
        lmax = l if l > lmax
        i = i + l
    else
        i += 1
    end
end
puts "Longest period with more than #{x} degrees is of length #{lmax}"
end

An alternative solution is

def maxperiod2( t, x )
    lmax = 0
    l = 0
    t.each{ |v|
        if v > x
            l += 1
        else
            lmax = [l,lmax].max
            l = 0
        end
    }
    puts "Longest period with more than #{x} degrees is of length #{lmax}"
end

Solution 4

% Decode a run-length encoded list
decode( [], [] ).
decode( [(1,X)|Ys], [X|Zs] ) :-
    decode( Ys, Zs ).
decode( [(N,X)|Ys], [X|Zs] ) :-
    N > 1,
    N1 is N - 1,
    decode( [(N1,X)|Ys], Zs ).

% Alternative solution with 2 predicates
decode2( [], [] ).
decode2([X|Xs], Z) :-
    decode3(X, Z1),
    decode2(Xs, Z2),
    append(Z1, Z2, Z).

decode3( (1,X), [X] ).
decode3( (N,X), [X|Xs] ) :-
    N > 1,
    N1 is N - 1,
    decode3( (N1,X), Xs ).

Solution 5

pythagoras(A, B, C) :-
    between(1, C, A),
    between(1, C, B),
    X is C * C,
    Y is A * A + B * B,
    X = Y.

Solution 6

rotate :: [a] -> Int -> [a]
rotate [] _ = []
rotate xs 0 = xs
rotate (x:xs) n = rotate (xs ++ [x]) (n-1)

Solution 7

module List (split)
    split
  ) where

    --
    -- Split input list at a given position into 2 lists
    --
    split :: [a] -> Int -> ([a], [a])
split xs n |
    | n < 0
    | n > length xs =
    | otherwise = split2([], xs) n
split2 :: ([a], [a]) -> Int -> ([a], [a])
split2 (xs, ys) 0 = (xs, ys)
split2 (xs, (y:ys)) n = split2 (xs ++ [y], ys) (n - 1)

Solution 8
-module(ballot).
-export([loop/0]).

loop() ->
  loop( 0, 0 )
end.

loop( PosCounter, NegCounter ) ->
  receive
    "info" ->
      observer ! { PosCounter, NegCounter },
      io:format("PosCounter ~p\n", [PosCounter]),
      io:format("NegCounter ~p\n", [NegCounter]),
      loop( PosCounter, NegCounter );
    "yes" ->
      loop( PosCounter + 1, NegCounter );
    "no" -> loop( PosCounter, NegCounter + 1 );
    "bye" ->
      io:format("Bye\n")
  end.

Pid = spawn(fun ballot:loop).