Programming Paradigms
Written Exam (6 CPs)

31.01.2017

First name		Last name	
Student number		Signature	

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

Question	Marks	Achieved
1	24	
2	12	
3	8	
4	10	
5	10	
6	10	
7	12	
8	14	
Total	100	

Exercise 1 (24 marks)

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

b. (4 marks) Given is the expression `[20, "20.0", 20.0]`.

   - Is the expression legal in Ruby?
   - Is the expression legal in Haskell?

   Explain your answers and briefly describe the meaning of this expression.

c. (4 marks) Briefly describe the difference between the following expressions in Prolog:

   - `?- 7 = 3 + 4`
   - `?- 7 is 3 + 4`

   What will be the result of executing these expressions?

d. (4 marks) What is wrong with the following case statement in Erlang? How could you fix the code?

```
case X of
   {_,_} -> doA;
   {_,3} -> doB;
   {2,_} -> doC;
   {2,3} -> doD
end.
```

e. (4 marks) Briefly explain the concept of list comprehension in Haskell and give a short example.

f. (4 marks) Consider the Haskell function `prod x y = x * y`. How is the function call `prod 2 4` evaluated? What is the name of this evaluation concept?

Exercise 2 (12 marks) Assume temperature data stored in an array `t`. Write a Ruby function `maxperiod(t,x)` that calculates the length of the longest (hot) period with temperature values 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 3 (8 marks) Write a Ruby function `palindrome(a)` to check whether an array of characters represents a palindrome, i.e., is identical to the reversed array. For instance, the array `['A', 'B', 'B', 'A']` represents a palindrom, while `['A', 'B', 'B', 'A', 'C']` does not. You are not allowed to use Ruby's `array` method `reverse`.

Exercise 4 (10 marks) The product of two natural numbers can be expressed as a repeated addition using the following recursive definition (Peano axioms):

$$\begin{aligned} 0 * y &= 0 \\ x * y &= (x-1) * y + y \end{aligned}$$

Write a Prolog program to compute a product using this definition.

Exercise 5 (10 marks) Write a Prolog program `subst(X,L1,Y,L2)` that replaces all occurrences of `X` in list `L1` by two occurrences of `Y` to obtain list `L2`. For example, `subst(a,[a,b,c],1,L2)` instantiates `L2=[1,1,b,c]`.

Exercise 6 (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, a 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)

Exercise 7 (12 marks) Write a Haskell function `diffrev`, which takes as input two lists, A and B, each containing n numbers, and produces in output a list C such that

$$\begin{aligned} C[0] &= A[0] - B[n-1] \\ C[1] &= A[1] - B[n-2] \\ &\dots \\ C[n-1] &= A[n-1] - B[0] \end{aligned}$$

For example, `diffrev [1,2,10] [2,5,-2]` returns `[3,-3,8]`. (Hint: think about breaking the problem down in more than one function!)

Exercise 8 (14 marks) Write a Haskell module that exports a tail-recursive function `noOfElem` that takes as input an element `x` and a list and returns the number of occurences of `x` in the list. For instance, `noOfElem 1 [1,2,3,1]` returns 2.

Solution 1

a. 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.

b. This is an array containing an integer, a string, and a float. Yes, it is legal in Ruby to mix different types in the same array.

   This is a list containing an integer, a string, and a float. No, in Haskell it is not allowed to store different types in the same list.

c. In the expression `7 = 3 + 4`, the unification operator `=` tries to match the left-hand and right-hand side. This is not possible here, since `3+4` is not evaluated, hence the two sides are different, and the goal fails.
   In the expression `7 is 3 + 4`, the `is` operator first evaluates the right-hand side to the value `7` and then matches it to the left-hand side; the goal succeeds.

d. The cases are in the wrong order, i.e., the most general ones come first. Consequently, the later cases will never be reached. The code can be fixed by reversing the order of the cases.

e. List comprehension in Haskell is a compact way to specify complex and possibly infinite lists by specifying an output function, a variable, an input set and a predicate. Example: The list of even numbers between 1 and 100 can be defined as
   `[x | x <- [1..100], mod x 2 == 0]`

f. The function is evaluated in two steps:

   - The first input parameter is applied, i.e., `prod 2`, yielding a partially evaluated function `(\y -> 2 * y)`
   - The partially evaluated function is applied to the second argument, i.e., `(\y -> 2 * y) 4`, yielding `8`

   This is called curried functions.

Solution 2

```
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 3

```
def palindrome(a)
  if a.length == 1 || a.length == 0
    true
  else
    if a[0] == a[-1]
      palindrome(a[1..-2])
    else
      false
    end
  end
end
```

Solution 4

```
prod( 0, _, 0 ).
prod( N, M, P ) :-
    N > 0,
    N1 is N - 1,
    prod( N1, M, K),
    P is K + M.
```

Solution 5

```
subst(_, [], _, []).
subst(X, [X|L], Y, [Y,Y|M]) :-
    !,
    subst(X, L, Y, M).
subst(X, [Z|L], Y, [Z|M]) :-
    subst(X, L, Y, M).
```

Solution 6

```
-module(sign).
-export([loop/0]).

loop() ->
  receive
    N when is_number(N) ->
      if
        N < 0 -> io:format("Number is negative~n");
        N > 0 -> io:format("Number is positive~n");
        N == 0 -> io:format("Number is zero~n")
       end,
      loop();
    "bye" ->
      io:format("Bye~n");
    _ ->
      io:format("Unexpected message~n"),
      loop()
  end.

P = spawn( fun sign:loop/0 ).
```

Solution 7

```
module Diff (diffrev) where

rev :: [Integer] -> [Integer]
rev [] = []
rev (h:t) = rev(t)++[h]

diff :: [Integer] -> [Integer] -> [Integer]
diff [] [] = []
diff (h1:t1) (h2:t2) = [h1-h2]++(diff t1 t2)

diffrev :: [Integer] -> [Integer] -> [Integer]
diffrev a b = diff a (rev b)
```

Solution 8

```
module NoOfElem (noOfElem) where

noOfEl ::  Eq a => a -> [a] -> Int -> Int
noOfEl x [] cnt = cnt
noOfEl x (h:t) cnt | x == h = noOfEl x t (cnt+1)
noOfEl x (h:t) cnt = noOfEl x t cnt

noOfElem ::  Eq a => a -> [a] -> Int
noOfElem x lst = noOfElem x lst 0
```

Non tail-recursive version:

```
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
```