Programming Paradigms
Written Exam

17.06.2014

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	25	
2	10	
3	10	
4	10	
5	10	
6	10	
7	15	
8	10	
Total	100	

Exercise 1 (25 marks)

a. (5 marks) Briefly describe the main differences between a compiled language and an interpreted language.

b. (5 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.

c. (5 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. (5 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.

e. (5 marks) Briefly explain the concept of list comprehension in Haskell and give a short example.

Exercise 2 (10 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 3 (10 marks) Write a Ruby function to calculate the fuel consumption of a car (in liters) for road trips it has taken. The input parameters of the function are `fuelConsumpt`, measured in liters per 100 kilometers, and `trips[]`, an array containing the distances of road trips (in kilometers) taken by that car. Compute the overall fuel consumption for the voyages stored in `trips`.

Exercise 4 (10 marks) Write a Prolog program that removes all duplicates from a list and returns the duplicate-free list. For instance, given the list `L1 = [a,b,a,c,a,a,b]`, the duplicate-free list is `L2 = [c,a,b]`. Note that the order of the elements in `L2` does not matter. You can use the built-in predicate `member(X,L)` to check whether `X` is a member in `L`.

Exercise 5 (10 marks) Write a Prolog program that finds all solutions to the equation

$$x \cdot x = y + y$$

with x being in the range of $[1, 3]$ and y in the range of $[1, 5]$.

Exercise 6 (10 marks) Write a function `loop` for an Erlang process that does the following. When it receives the message `"event"`, it sends a message containing the current count of events to a process registered with an atom called `observer`, displays the count on the screen, and increments the current event count. If it receives the message `"bye"`, it displays "Bye" on the screen and quits.

Exercise 7 (15 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. An empty string is balanced. For example,

- "(())" is balanced
- "(())()" is balanced
- ")(()" is not balanced
- "())" is not balanced

Exercise 8 (10 marks) Look at the following recursive Haskell program.

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

```
mystery :: [a] -> Integer
mystery [] = 0
mystery (h:t) = 1 + mystery t
```

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

Solution 1

a. In a compiled language, a program (the source code) is translated into a form that can be run directly on a computer's processor.
Interpreted languages are processed by a higher-level virtual machine. Usually, a program is translated on the fly, i.e., a statement is tanslated and the immediately executed.

b. No, Erlang programs don't have to be modified to run on a multi-core. The main reason for is are 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.

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 expression is legal in Ruby and represents an array of an integer, a string, and a float. Ruby allows arrays of objects with different type. This is also called Duck typing.
The expression is not legal in Haskell. Haskell is a strongly typed language and does not allow an array to contain elements with different type.

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]`

Solution 2

```
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 3

```
def fuel(fuelConsumpt, trips=[])
  sum = 0.0
  trips.each{ |x| sum = sum + (x/100 * fuelConsumpt) }
  return sum
end
```

Solution 4

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

Solution 5

```
x(1).
x(2).
x(3).
y(1).
y(2).
y(3).
y(4).
y(5).
eq(X,Y) :- x(X), y(Y), A is X*X, A is Y+Y.
```

Solution 6

```
-module(eventcount).
-export([loop/1]).

loop(Counter) ->
  receive
    "event" ->
      observer ! Counter,
      io:format("event ~p~n",[Counter]),
      loop(Counter+1);
    "bye" ->
      io:format("bye~n")
  end.
```

Solution 7

```
module Paran where

isbalanced :: [Char] -> Int -> Bool
isbalanced [] 0 = True
isbalanced [] nonzero = False
isbalanced (h:t) x =
   if x < 0 then
      False
   else
      if h == '(' then
         isbalanced t (x+1)
      else
         isbalanced t (x-1)
```

Solution 8

a. The program computes the length of an array.

b. Tail-recursive version: pass the length of the list seen so far as a second parameter; when the list is empty, the second parameter contains the length of the list.

```
module Tail (
  len
) where

len :: [a] -> Integer -> Integer
len [] x = x
len (h:t) x = len t (x+1)
```