Programming Paradigms
Written Exam

02.02.2015

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	15	
7	10	
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) 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. (5 marks) Consider the following Prolog program:

```
a(1).
a(z).
b(3).
b(z).
c(A,B) :- b(B), !, a(A).
d(A,B) :- b(B), a(A).
```

   How many solutions are produced by the each of the two queries `c(A,B)` and `d(A,B)`? Briefly explain your answer.

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

e. (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.

Exercise 2 (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 3 (10 marks) Write a Ruby function `reverse(a, b)` that returns `true` if the integer array `a` is the reverse of the integer array `b`, and `false` otherwise. For example, `reverse([1,2,3,4],[4,3,2,1])` returns `true`, while `reverse([1,2,3],[4,3,2,1])` gives `false`. You are not allowed to use Ruby’s `array` method `reverse`.

Exercise 4 (10 marks) The knowledge base of a Prolog program describes an electrical supply grid:

```
supplies(p,t1).
supplies(p,t2).
supplies(t1,t3).
supplies(t1,t4).
supplies(t2,home1).
supplies(t2,home2).
supplies(t3,home3).
supplies(t4,home4).
supplies(t4,home5).
```

`p` is the power plant, `ti` stands for transformer i, and `homej` for home j. The clauses above indicate that there are power lines running between two different points. Assume that this is a hierarchical structure, i.e., there are no cycles in the grid.

Write rules for the predicate `hookedup(Y)` that tells whether `Y` is (directly or indirectly) connected to the power plant `p`.

Exercise 5 (10 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 will test 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).
```

will generate 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 (15 marks) Write an Erlang server that computes the Euclidean distance between two points in the plane, i.e., when receives two coordinates, it computes the distance between the points and sends it back.

The server process receives a tuple containing the process 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.

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

Exercise 7 (10 marks) Write a function in Haskell that finds an element in a list. It returns `True` if the element is in the list, otherwise it returns `False`.

```
module Findelement where

findelement ::
```

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. Compiled languages are translated into a from that can be run directly on a computer's processor. Usually the whole program is translated before it is run.

   Interpreted languages are processed by a higher-level virtual machine. Usually a program is translated on the fly, i.e., a statement is translated and then immediately executed.

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. (a) `c(A,B)` has 2 solutions: `A = 1, B = 3; A = z, B = 3.`

   The `cut` operator (`!`) stops backtracking and prevents `b(B)` from being matched with the second `b`-fact.

   (b) `d(A,B)` has 4 solutions: `A = 1, B = 3; A = z, B = 3; A = 1, B = z; A = z, B = z (or A = B, B = z).`

d. 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]`

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

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

Solution 3

```
def reverse( a, b )
  if  a.length == 0 && b.length == 0
    return true
  else
    if a[0] == b[-1]
      return reverse( a[1..-1], b[0..-2] )
    else
      return false
    end
  end
end
```

Alternative solution:

```
def reverse2( a, b )
  return true if a.length == 0 && b.length == 0 ||
                 a[0] == b[-1] && reverse( a[1..-1], b[0..-2] )
  return false
end
```

Solution 4

```
hookedup(Y) :- supplies(p,Y).
hookedup(Y) :- supplies(X,Y),hookedup(X).
```

Solution 5

```
square( S ) :-
  between( 0, S, X ),
  S is X * X,
  !.
```

Solution 6

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

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

Solution 7

```
module Findelement where

findelement ::  Eq a => a -> [a] -> Bool
findelement x [] = False
findelement x (h:t) =
   if x == h then
      True
   else
      findelement x t
```

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)
```