Prof. O. Nierstrasz
Boris Spasojević, Jan Kurš

Solution Exam Programming Languages

Date: Friday, 03.06.2016.
Duration: 70 minutes
Material: You are NOT allowed to use any material (e.g., script, exercises including solutions, notes, electronic devices...)
Number of exercises: 6
Total points: 80

Firstname, lastname: ________________________________

Matrikel: ________________________________

Put your name on each extra page you deliver.
Consecutively number all pages. Total number of extra pages: ________

Exercise 1 (20 Points)

Answer the following questions (do not write more than 3 sentences):

1. What is a tail-recursive function and what is the advantage compared to a non-tail-recursive function?
   Answer:

   Tail-recursive means calling itself is the last thing the function does. The advantage is reusing the stack.

2. What is a higher-order function? Give an example.
   Answer:

   A higher-order function is a function that takes another function as an argument or returns a function. Example: the `map` function.

3. What is the difference between a static and a dynamic type? Give an example.
   Answer:

   The static type of a variable or expression is a type which can be determined by the type inference system solely based on the program code, thus at compile time. In contrast, the dynamic type of a variable cannot be determined in such a way because the variable may take on different values at run time.
   Example:
   `Object x = new Vector();`
   the static type of `x` is `Object`, the dynamic type is `Vector`.

4. What are the operations one can perform in lambda calculus?
   Answer:

   Alpha conversion, betta reduction, eta reduction.

5. What is a fixed point of a function?
   Answer:

   A fixed point of a function `f` is a value `p` such that `f p = p`.

6. What is the difference between syntax and semantics?
Answer:

Syntax considers the arrangement of words and phrases to create well-formed sentences in a language.
Semantics considers the meaning of a word, phrase, sentence or text.
You can create well-formed sentences (according to the syntax) that don't have a meaning (according to semantics).

7. What is the Principle of Substitutability?
Answer:

Principle of Substitutability means that an instance of a subtype can always be used in any context in which an instance of a supertype was expected.

8. How does Java supports subtyping? How does it support specializations?
Answer:

Subtyping *is supported by the usage of interfaces and inheritance.*
Specialization *is supported by inheritance.*

9. Where do you in Javascript define properties that are shared between a group of objects (i.e., static members in Java)?
Answer:

Either using the same prototype and defining on the prototype side or in a global scope.

10. How would you define a logical negation operator `neg(X)` in Prolog?
Answer:

`neg(X) :- X, !, fail.  neg(_).`

Exercise 2 (6 Points)

A very junior programmer wanted to write the “hello world” program in Postscript. Here is the result:

```
/Times-Roman findfont
18 scalefont
setfont
100 500 moveto

/myprint {
    exch
    /mystring exch def
    show mystring show
} def

/mystring (Hello) def
mystring
/mystring (World) def
mystring
myprint
showpage
```

1. Did the programmer succeed? If not explain what the output is and why. How can the program can be fixed to achieve the junior programmer goal?

Answer:

The result is `WorldHello` since `/myprint` swaps on the stack, redefines `mystring` as `Hello` (destroying it on the stack) and shows `World` then shows `mystring`.
It is better to write: `(Hello World) show`

Exercise 3 (18 Points)

A *triangular number* is the number of objects that can be arranged in a triangle as is shown in Figure 1. This is how bowling pins, pool balls, or snooker balls are arranged. The figure shows the first six triangular numbers: 1, 3, 6, 10, 15, 21.
The *tetrahedral number* represents the number of objects arranged in a pyramid (more precisely, a tetrahedron) built up from triangles as shown in the Figure.

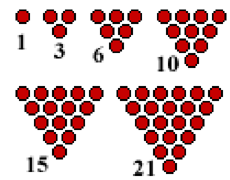


Figure 1: Triangular numbers.

The exercise:

1. Write a function to calculate the n-th triangular number in Haskell.
2. Write a function to calculate the n-th tetrahedral number in Haskell.
3. Infer the type of the following function and explain your steps. Is the function monomorphic or polymorphic?

```
myMap f [] = f
myMap f (x:xs) = myMap (x (f+1)) xs
```

Answer:

```
triangular 0            = 0
triangular n            = n + triangular (n-1)

tetrahedral 0        = 0
tetrahedral n        = (triangular n) + (tetrahedral (n-1))

myMap :: Num a => a -> [a -> a] -> a
```

Exercise 4 (12 Points)

1. Consider the following λ-expressions. Indicate which occurrences of variables are bound and which ones are free in the expressions.

   (a) `(λ a b . c a b) a b (λ c d . d c)(λ e f . f)`

   (b) `((λ u v . λ w . w (λ x . x(u)) (v)) (y)) (λ z . λ y . z(y))`

   (c) `(λ y . λ x . z(x(λ x . y(z)))) (λ z . y(x(z)))`

2. Reduce the following λ-expression to its normal form if possible

   `(λ x y . y x) (λ x y . y x)(λ x . x x)(λ y . y)`

Answer:

```
(λ a b . c a b) a b (λ c d . d c)(λ e f . f)
(λ a b . f b b) f f (λ c d . b b)(λ e f . b)

((λ u v . λ w . w (λ x . x(u)) (v)) (y)) (λ z . λ y . z(y))
((λ u v . λ w . b (λ x . b(b)) (b)) (f)) (λ z . λ y . b(b))

(λ y . λ x . z(x(λ x . y(z)))) (λ z . y(x(z)))
(λ y . λ x . f(b(λ x . b(f)))) (λ z . f(f(b)))


(λ x y . y x) (λ x y . y x)(λ x . x x)(λ y . y)
(λ x . x x)(λ x y . y x)(λ y . y)
(λ x y . y x)(λ x y . y x)(λ y . y)
(λ y . y)(λ x y . y x)
(λ x y . y x)
```

Exercise 5 (12 Points)

Suppose you have a small JavaScript program with a database of people:

```
var alice = Object.create(person);
alice.name = "Alice";
alice.age = 22;

var bob = Object.create(person);
bob.name = "Bob";
bob.age = 29;

var cyril = Object.create(person);
cyril.name = "Bob";
cyril.age = 45;
```

1. What is the prototype of Alice, Bob and Cyril?
   Answer:

   Person

2. Lets say Cyril is a manager. A manager has to be responsible. How would make Cyril (only Cyril, not Alice and Bob) respond to the message `isResponsible`?
   Answer:

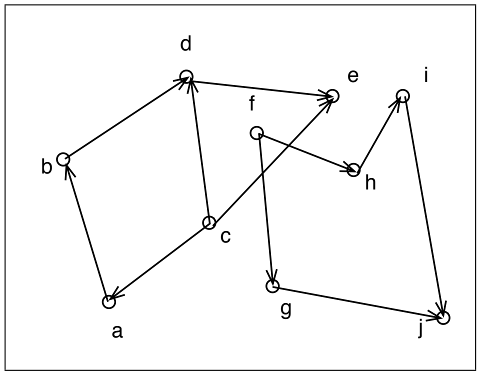
   ```
   manager.isResponsible = function() {
       alert("I am very responsible");
   }
   ```

3. Bob wants to be manager as well, so he tries to be responsible as well. How would you make Bob to become responsible?
   Answer:

   ```
   bob.isResponsible = manager.isResponsible.
   ```

Exercise 6 (12 Points)

Consider the following directed graph:



Write a Prolog database consisting of the following predicates:

a. `line(p1,p2)` that is true iff there is a direct connection from `p1` to `p2`.

b. `triangle(p1,p2,p3)` that is true iff `p1`, `p2`, and `p3` form a triangle (independent of the connection directions).

c. `quadrangle(p1,p2,p3,p4)` that is true iff `p1`, `p2`, `p3`, and `p4` form a quadrangle (independent of the connection directions).

d. `reachable(p1,p2)` that is true iff there is a directed path from `p1` to `p2` (i.e. iff `p2` is reachable from `p1` respecting the connection directions).

Answer:

```
connection(A,B) :- line(A,B); line(B,A).
in(X,[X|_]).
in(X,[_|L]):- in(X,L).

line(a,b).
line(b,d).
line(c,a).
line(c,d).
line(c,e).
line(d,e).
line(f,g).
line(f,h).
line(g,j).
line(h,i).
line(i,j).
```

```
triangle(A,B,C) :- connection(A,B), connection(B,C), connection(C,A).


quadrangle(A,B,C,D) :- connection(A,B), connection(B,C),
                        connection(C,D), connection(D,A),
                        not(in(D,[A,B,C])).

reachable(X,Y) :- line(X,Y).
reachable(X,Y) :- line(X,Z), reachable(Z,Y).
```

Points

Exercise 1

Task	Points	Score
1	2	
2	2	
3	2	
4	2	
5	2	
6	2	
7	2	
8	2	
9	2	
10	2	
Total	**20**	

Exercise 2

Task	Points	Score
1	6	
Total	**6**	

Exercise 3

Task	Points	Score
1	6	
2	6	
2	6	
Total	**18**	

Exercise 4

Task	Points	Score
1(a)	2	
1(b)	2	
1(c)	2	
2	6	
Total	**12**	

Exercise 5

Task	Points	Score
1	3	
2	3	
3	3	
4	3	
Total	**12**	

Exercise 6

Task	Points	Score
a	3	
b	3	
c	3	
d	3	
Total	**12**	

TOTAL

Exercise	Points	Score
1	20	
2	6	
3	18	
4	12	
5	12	
6	12	
Total	**80**	